The Industrial Internet Reference Architecture

Version 1.10

An Industry IoT Consortium Foundational Document

2022-11-07

CONTENTS

FIGURES

TABLES

This technical report continues the work of the Industrial Internet Consortium (IIC) by refining this *Industrial Internet Reference Architecture (IIRA)*, as an update since initial publication in July 2015. These updates reflect new technologies, concepts and applications emerging in IIoT and they clarify existing concepts and descriptions when appropriate. It provides guidance to IIoT architects, business leaders, implementers and users at every level to optimize their endeavors in establishing IIoT systems,[1] consummating the convergence of operational technology (OT) and information technology (IT) to achieve the tremendous economic benefits IIoT has to offer.

This technical report describes the Industrial Internet Reference Architecture (IIRA) for Industrial Internet of Things (IIoT) systems. It specifies architectural concerns, constructs and approaches to aid in development, documentation, and communication of IIoT systems. The reference architecture uses a common vocabulary and a standard-based framework to describe business, usage, functional and implementation viewpoints.

This IIRA has two primary purposes. For IIC work efforts, it is the foundational framework for other technical documents. For the broader IoT community, it provides guidance and assistance in the development, documentation, communication and deployment of IIoT systems.

The IIRA presents an architectural description of IIoT systems using *ISO/IEC/IEEE 42010:2011* [IEEE-42010] architecture concepts. This document thinks ahead to include technology concepts that the IIC is experimenting with through its testbed programs.

This document is organized as follows:

- *Chapter 1*: Reference Architecture Concepts
- *Chapter 2*: Architecture Framework
- *Chapter 3*: Business View
- *Chapter 4*: Usage View
- *Chapter 5*: Functional View
- *Chapter 6*: Implementation View

This document is primarily for IIoT system architects. We assume the reader is familiar with general architecture concepts, architecture frameworks and reference architectures. It can also be used by, and provides value for, plant managers, IT managers, business managers and others who want to understand better how the convergence of OT and IT is an important part of improving their business.

System architects can use this IIRA systematically as an architectural template to define their IIoT system requirements and design concrete architectures to address them. Using this common

[1] An IIoT System is system where the components are connected via a digital network and one or more of those components interact with the physical world, as defined in the IIC Vocabulary [IIC-IIV].

approach to architecture design assists in consistent architecture implementation across different use cases while meeting system requirements. It also assists in achieving a common understanding of the system among its diverse stakeholders, which will aid in system deployment and enhance system interoperability across industrial sectors.

1 INDUSTRIAL INTERNET REFERENCE ARCHITECTURE CONCEPTS

A reference architecture provides guidance for the development of system, solution and application architectures. It provides common and consistent definitions for the system of interest, its decompositions and design patterns, and a common vocabulary with which to discuss the specification of implementations and compare options.


Example

A reference architecture for a residential house states that all residential houses need to provide one or more bedrooms, bathrooms, a kitchen and a living area. This set of rooms is accessible inside the house through doors, hallways, and stairways, and from outside through a main and a back door. The house provides a safe environment against threats such as fire, hurricanes and earthquakes. The structure of the house needs to sustain snow and wind load that may be found in its local environment. The house needs to provide reasonable measures to detect and prevent unauthorized intrusions.

A reference architecture provides a common framework for more detailed discussions. By staying at a higher level of abstraction, it enables the identification and comprehension of the most important issues and patterns across its applications in many different use cases. By avoiding specifics, a reference architecture allows subsequent designs to follow the reference architecture without the encumbrance of unnecessary and arbitrary restrictions.

1.1 INDUSTRIAL INTERNET REFERENCE ARCHITECTURE

The IIRA is a standards-based open architecture for IIoT systems. The IIRA maximizes its value by having broad industry applicability to enhance common understanding, drive interoperability, to map applicable technologies and guide technology and standard development. The architecture description and representation are generic and at a high level of abstraction to support the requisite broad industry applicability. The IIRA distills and abstracts common characteristics, features and patterns from use cases defined in the IIC and elsewhere. It will be refined continually as feedback is gathered from its application in the testbeds developed in IIC and real-world deployments. The IIRA is also intended to transcend today's available technologies and so can identify technology gaps based on the architectural requirements. This will in turn drive new technology development efforts by the IIoT community.

2 BACKGROUND ON REFERENCE ARCHITECTURES

Many stakeholders are involved when considering complex IIoT systems. They have many intertwining concerns pertinent to the system of interest, covering the full lifecycle of the system. System complexity requires a framework to identify and classify stakeholder concerns into appropriate categories. Such a framework allows a systematic evaluation of such systems, and the resolution necessary to architect and build such systems.

To address this need, the Industry IoT Consortium used the *ISO/IEC/IEEE 42010:2011 Systems and Software Engineering–Architecture Description* [IEEE-42010] standard to define its Industrial Internet Reference Architecture. standard codifies architecting conventions and common practices and provides an ontology for the description of architectures and architecture frameworks. Taken from ISO/IEC/IEEE 42010:2011, Figure 1.1-1 expresses the relations between the terms and concepts of an architecture and architecture framework.

A reference architecture is a type of architecture description and identifies conventions, principles and practices for consistent IIoT architectures. This standard-based reference architecture facilitates easier evaluation, and systematic and effective resolution of stakeholder concerns. It serves as a valuable resource to guide the development and the documentation of, and the communication about IIoT systems.

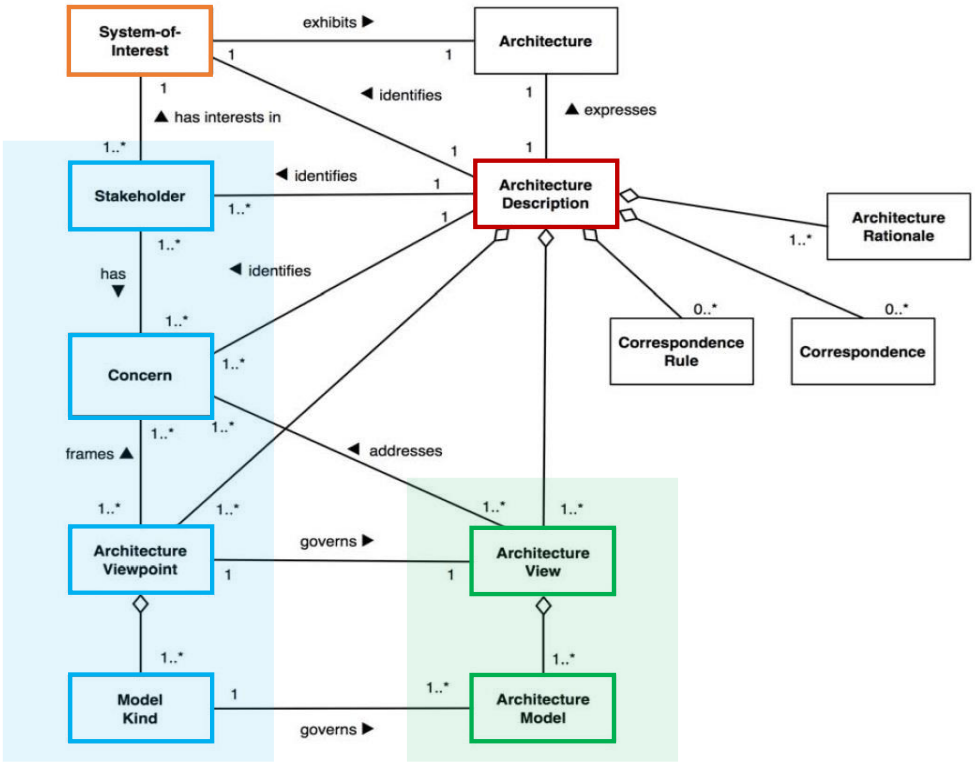


Figure 1.1-1: ISO/IEC/IEEE 42010:2011—Architecture Description[1]

2.1 Reference Architecture (a Type of Architecture Description)

A reference architecture is a type of architecture description that provides a set of constraints and guidance based on a set of related systems. A reference architecture contains information identifying the fundamental architecture constructs and specifies concerns, stakeholders, viewpoints, model kinds, correspondence rules and conditions of applicability. System architects can use a reference architecture to discover, describe and organize topics of interest (concerns) about the system at hand; they can further use architecture viewpoints and views to clarify, analyze and resolve these concerns.

At the core of the ISO/IEC/IEEE Architecture Description standard are viewpoints and views. A *viewpoint* comprises conventions framing the description and analysis of specific system concerns. A viewpoint frames one or more concerns. The term *concern* refers to any topic of interest pertaining to the system. A *stakeholder* is an individual, team, organization or classes thereof, having an interest in a concern and by extension an interest in the viewpoint and

[1] The colored highlights are added by this document to indicate the architectural constructs that are described by IIRA.

system.[1] To aid in describing, analyzing and resolving concerns, one or more modeling constructs can be defined as the *model kinds*[2] for each viewpoint.

Architecture *view* work product expresses the architecture of a system from the perspective of specific system concerns. Following the approach defined by the ISO/IEC/IEEE Architecture Description standard, the description, analysis and solution of the set of specific concerns in each of the viewpoints are expressed as architecture views for each viewpoint. Applying the model kinds defined in each viewpoint to describe, analyze and resolve the concerns consequently result in the creation of architecture models that make up the respective architecture views. Together, the architecture views with their architecture models represent the architecture.


Example

A common approach for designing a complex system is to decompose it into constituent subsystems. Suppose we want to address the concerns of what the functional subsystems are, their interfaces and how they interact to realize the desired system behaviors. A functional decomposition of the system can make each of the subsystems easier to conceive, understand, design, implement, reuse and maintain. A component diagram may be used to describe the structure of the subsystems and their interfaces; sequence diagrams the way in which the subsystems interact; and state diagrams the way in which the system or one of its subsystems behaves in response to external events. These diagrams and their associated documentation describe the concerns of the functional decomposition. The component, sequence and state diagrams are the model kinds that address the concerns of functional structure of the system. These *model kinds* can be applied to analyze the system of interest. The resultant concrete models become part of the architecture models.[3]

2.2 INDUSTRIAL INTERNET OF THINGS REFERENCE ARCHITECTURE

The IIRA adopts the general concepts and constructs in the ISO/IEC/IEEE architecture description specification, specifically, *concern, stakeholder, view* and *viewpoint* as its architecture frame, and *views* and *models* as its architecture representation in describing and analyzing on important common architecture concerns for IIoT systems.

The IIRA describes the architecture for its intended class of systems of interest: IIoT systems. It highlights the important architectural concerns commonly found in IIoT systems across industrial

[1] An IIoT system may become a stakeholder of itself as it becomes intelligent, capable of learning and making decisions itself as an autonomous agent.

[2] Per ISO/IEC/IEEE 42010, a model kind captures conventions for a type of model. In an analogue to object-oriented programming, it is similar to a class of model that can be instantiated to represent actual business objects.

[3] There are other architecture models addressing other concerns.

sectors and classifies them into viewpoints along with their respective stakeholders. It then describes, analyzes and, where appropriate, provides guidance to resolve these concerns in these viewpoints.

Figure 2.2-1 illustrates the key ideas in the IIRA and its application.

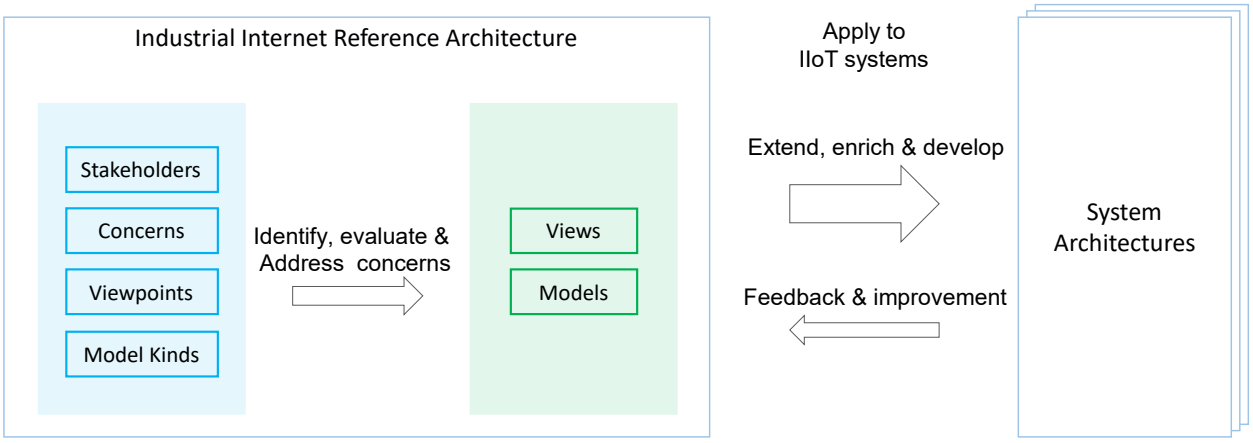


Figure 2.2-1: IIRA Constructs and Application

The IIRA is at a level of abstraction that excludes architectural elements whose evaluation requires specificities only available in concrete systems.

Within the IIRA, the models in the views are chosen because they address the respective concerns at the appropriate level of abstraction for the view and demonstrate the key ideas of this reference architecture. They are not, however, the sole models and views for addressing concerns in the viewpoints, nor are they at a depth sufficient to implement a real system. The views can be used as a starting point for concrete architecting, assisting in the construction of an abstract architecture that addresses concerns extended and enriched, with specific use case requirements in accordance with the needs of the specific IIoT system at hand.

System architects interested in following the ISO/IEC/IEEE 42010 Architecture Description to develop their concrete architectures can first apply IIRA as its base framework, extend and enrich the constructs provided in the IIRA based on their specific system requirements where necessary, and develop those constructs not described in IIRA as part of the architecting process.

The following sections define the Industrial Internet Architecture stakeholders and identify their key concerns. The views addressing these concerns are then described in more details in Chapters 3 through 6.

2.3 USING THE IIRA

The purpose of the IIRA is to provide guidance to system architects to assist the architects in building IIoT systems. The IIRA v2.0 has been designed to improve the user experience (and increase the value provided) by addressing the stakeholder concerns more clearly. By providing four architecture views the system architect is guided through a development flow focusing on

the unique aspects of IoT. The flow starts with a conceptual view, the business view, user view, functional view and finally implementation view. Each view builds on the last view and help guide the reader as they go from conceptualization all the way to implementation.

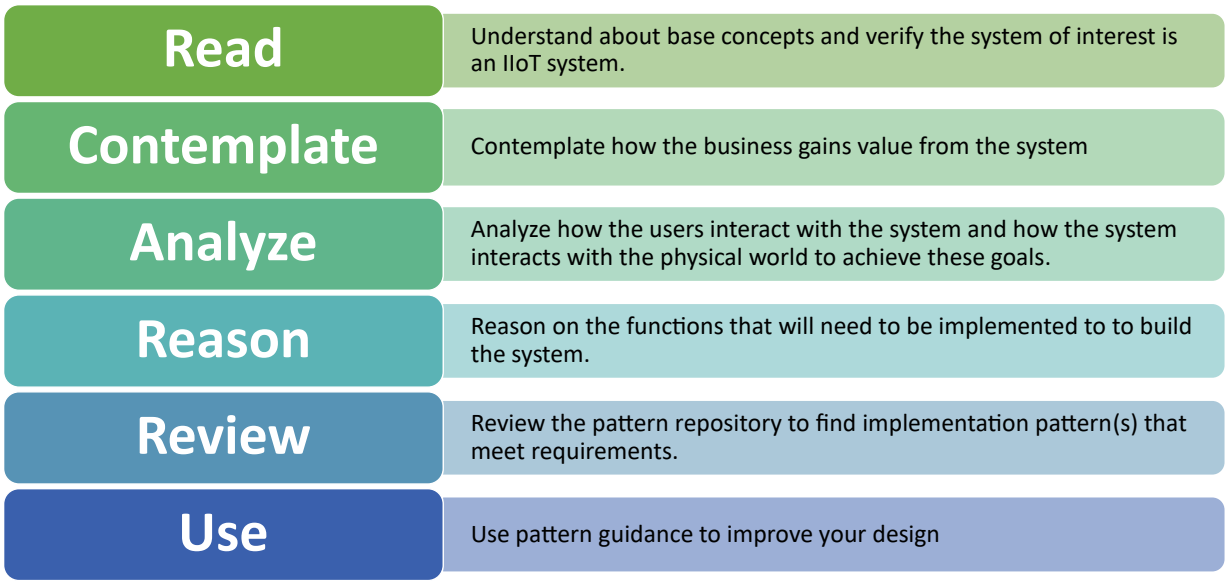


Figure 2.3-1: How to use IIRA

2.4 Industrial Internet Stakeholders

There are many different stakeholders who have concerns related to IIoT. These include:

Product managers that manage the conception, creation, execution, and lifecycle of the IIoT system and services. They are involved in the specification of the IIoT system under consideration and represent the users in its ultimate usage.

System engineers implement and maintain the software and hardware components that comprise the system.

System architects create the structural design of the software and hardware components that comprise the IIoT system.

Component architects design the individual elements (system components) that comprise the IIoT system.

Developers create the IIoT system from the given specifications.

Integrators deploy the IIoT system for specific application scenario and integrate it with other relevant systems to make IIoT system operational.

System operators maintain the IIoT system and are responsible for the daily tasks necessary to keep the system functioning.

The concerns of these stakeholders are addressed in the viewpoints described in the following section.

2.5 Industrial Internet Reference Architecture Viewpoints

The IIRA viewpoints are defined by analyzing the various IIoT use cases developed by the IIC and elsewhere, identifying the relevant stakeholders of IIoT systems and determining the proper framing of concerns. These four viewpoints are:

- Business Viewpoint
- Usage Viewpoint
- Functional Viewpoint
- Implementation Viewpoint

As shown in Figure 2.5-1, these four viewpoints form the basis for the four views that address the IIoT system concerns. Architects can then use these industrial internet views as the basis of their architecture and may extend them by defining additional views and viewpoints as needed to organize system concerns based on their specific system requirements.

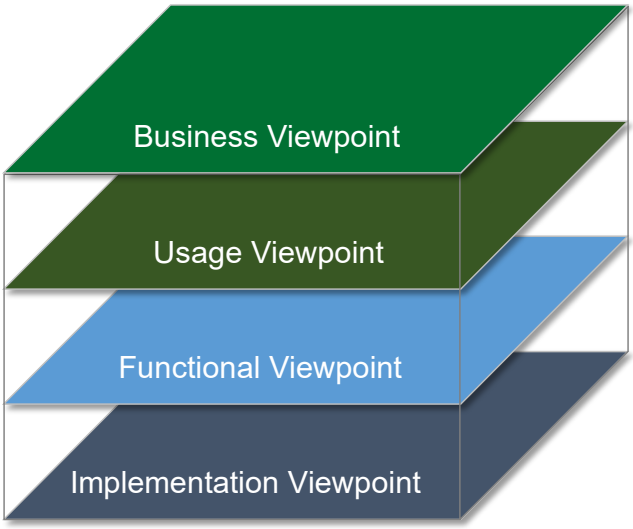


Figure 2.5-1: Industrial Internet Architecture Viewpoints

2.5.1 Business Viewpoint

The *business viewpoint* frames the concerns of the business stakeholders and their business vision, values and objectives in establishing an IIoT system in its business and regulatory context. It further identifies the objectives of the IIoT system.

The concerns for the business viewpoint are business-oriented and are of interest to business decision-makers, product managers and system engineers.

The business view model is of the narrative statement model kind that specifically addresses the business stakeholders' concerns. A *narrative statement* is a textual description of model artifacts.

Chapter 5 details the business view that corresponds to this viewpoint.

2.5.2 USAGE VIEWPOINT

The *usage viewpoint* frames the concerns of expected system usage. It is typically represented as sequences of activities involving human or logical (e.g. system or system components) users that deliver its intended functionality in achieving its fundamental system capabilities.

The stakeholders of these concerns typically consist of system engineers, product managers and others, including those involved in the specification of the IIoT system under consideration and those who use it.

The usage view contains models of context model kind. A context model depicts a system of interest, or other entity, in the context of its environment to identify external entities with which the system interacts. A context model is important to help delineate the system boundary (what's inside and what's outside) and interactions with external entities.

Chapter 6 details the usage view the corresponds to this viewpoint.

2.5.3 FUNCTIONAL VIEWPOINT

The *functional viewpoint* focuses on the functional components in an IIoT system, their structure and interrelations, the interfaces and interactions between them, and the relation and interactions of the system with external elements in the environment, to support the usages and activities of the overall system.

These concerns are of particular interest to system and component architects, developers and integrators.

The functional view contains two models, a functional domain model of concepts model kind, and a functional requirements model of narrative statement model kind. A concepts model depicts a set of concepts and their relationships for a domain of problem under consideration. It includes both diagrams and narrative descriptions. A narrative statement model provides text to be used as a part of one or more architecture views to address concerns.

Chapter 7 details the business functional view that corresponds to this viewpoint.

2.5.4 IMPLEMENTATION VIEWPOINT

The *implementation viewpoint* frames concerns with the technologies needed to implement functional components (functional view), their communication schemes and their lifecycle procedures. These elements are coordinated by activities (usage view) and support the system capabilities (business view).

These concerns are of interest to system and component architects, developers and integrators and system operators.

The patterns in the implementation view are of architecture pattern model kind. An architecture pattern model kind describes the structure of a system and are used in the construction of new systems. Architecture patterns are described using both common architecture frameworks and specific domain languages.

Chapter 8 details the Implementation View that corresponds to this viewpoint.

2.6 CROSSCUTTING CONCERNS, SYSTEM CHARACTERISTICS AND THEIR ASSURANCE

The business, usage, functional and implementation views facilitate a systematic way to identify IIoT system concerns and their stakeholders, to bring similar or related concerns together so they can be analyzed and addressed effectively. The deliberation of the concerns is often performed within each of the viewpoints to which they belong, but not always.

The order in which the business, usage, functional and implementation viewpoints are arranged, from top to bottom, as depicted in Figure 2.7-1, reflects a general interaction pattern between the viewpoints. Broadly speaking, decisions from a higher-level viewpoint guide and impose requirements on the viewpoints below it. For example, the decisions resulting from the business viewpoint has direct influence on the usage viewpoint and so forth. On the other hand, the deliberation of the concerns in a lower viewpoint, including implementing requirements from the viewpoints above it, validate and in some cases cause revisions to the analysis and decisions in the viewpoint above it. For example, deliberation in the usage viewpoint may validate whether the system capability proposed in the business viewpoint can be realized.

Moreover, there are classes of system concerns, such as those related to safety and security, which may require consistent consideration across the viewpoints. These are sometimes referred to as crosscutting concerns. This class of concerns is related to system properties resulting from its components and interactions among them—the emergent properties of the system. Emergent system-wide properties are called system characteristics. *System characteristics* are IIoT system properties and behaviors resulting from its constituent sub-systems, the nature of their interactions with each other and the context and the environment in which they operate. These properties usually have contractual value, e.g. a supporting Service Level Agreement (SLA), for the system stakeholders.

System concerns related to safety and security are of crucial importance to IIoT systems. To ensure the system is capable of demonstrating the expected system characteristics, it is essential to have a clear understanding of the business drivers for strong safety and security requirements and their potential effect on the business objectives in case they are not met. It also requires detailed consideration of how these requirements affects the usage of the system. The requirements and usage considerations must be reflected in the design of the functional components and the choice of technologies and actual implementation of the system.

Moreover, system characteristics are often subject to regulations, compliance requirements and contractual agreements and so need be measured and assessed. Because IIoT systems are built from multi-vendor components and solutions, possibly composed dynamically after deployment, it may be required to provide recorded claims and their supportive evidence of specific system characteristics in components to evaluate, select, acquire and assemble qualified components into the desired IIoT system.

More detailed discussions on the key crosscutting concern and their associated system characteristics, such as safety and security, and their assurance [OMG-SACM], including the concept of trustworthiness to address the entwined nature of security, safety, reliability, resilience, and privacy appropriately, can be found in *Key IIoT System Concerns* [IIC-KSC], *Industrial Internet Security Framework* [IIC-IISF], and elsewhere.

2.7 Scope of Applicability and Relationship to System Lifecycle Process

2.7.1 Scope of Applicability

The reference architecture purposely starts from a generic framework and seeks common architecture patterns to ensure wide applicability to industrial internet applications across industrial sectors. For this reason, this general framework stays at a high level in its architecture descriptions, and its concepts and models are at a high degree of abstraction. The application of this general architecture framework, as a reference architecture, to real-world usage scenarios transform and extend the abstract architectural concepts and models into detailed architectures addressing the specificity of the industrial internet usage scenarios, thereby guiding the next level of architecture and system design.

The IIC will evaluate feedback from practical implementations across industrial sectors, including the various IIC testbed initiatives, to ascertain the soundness and usefulness of IIRA in aiding the system-design process and may revise and improve this reference architecture as deemed necessary. IIC expects IIoT system implementers to identify additional common architecture patterns, especially those carrying next-level details in the architecture. From their feedback, IIC will document and make available additional architecture patterns where appropriate to aid in future system designs.

2.7.2 Relationship to System Lifecycle Process

The architecture concerns framed by this reference architecture may need to be addressed beyond the design phase of the system into its full lifecycle. This reference architecture, through its viewpoints, provides guidance to system-lifecycle processes from IIoT system conception, to design and implementation. Its viewpoints offer a framework to system designers to think iteratively through important common architectural issues in IIoT system creation. It also suggests some common approaches (concepts and models) as views in each of these viewpoints to aid the identification and resolution of important architectural issues. It is not a description of

a system-lifecycle process, which varies from one industrial sector to another, as that is out of scope. Rather, as shown in figure 3-6, this reference architecture is an architectural framework for system conceptualization and architecture highlighting important system concerns that may affect lifecycle process. IIoT system lifecycle processes and the expected use of this reference architecture in these lifecycle processes will be covered in a different set of IIC technical reports.

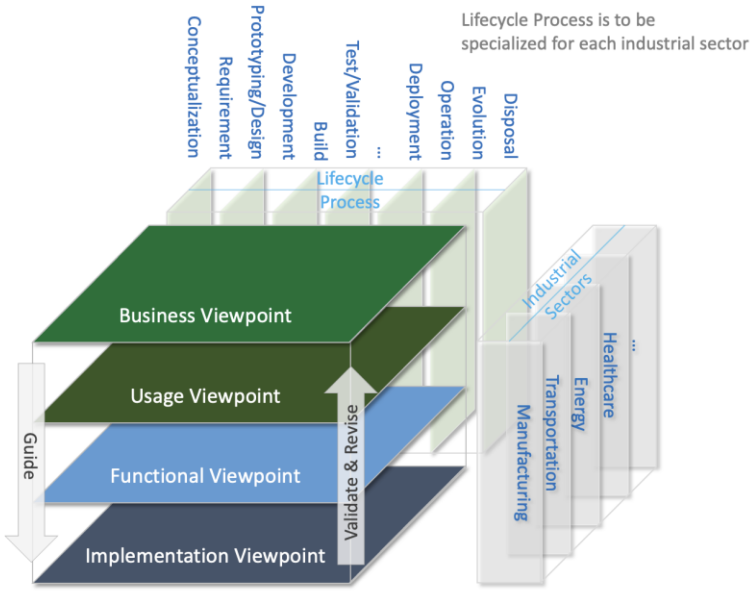


Figure 2.7-1: Relationship among IIRA Viewpoints

3 BUSINESS VIEW

Business view is an architecture view that addresses concerns related to the vision, values and objectives of the business stakeholders in establishing an industrial internet of things (IIoT) system in its business and regulatory context.

Business-oriented concerns such as business value, expected return on investment, cost of maintenance and product liability must be evaluated when considering an IIoT system as a solution to business problems. Moreover, an IIoT system must interact with one or more physical entity-of-interest either observing it, acting upon it or both. Figure 4-1 shows the concepts of the business viewpoint and the relationships between them.[1]

[1] This approach is based on the *Business Motivation Model* [OMG-BMM] by the Object Management Group (OMG), consistent with best practices in this domain. Some of the terminology has been changed to be consistent with the ISO/IEC/IEEE 42010:2011 [IEEE-42010].

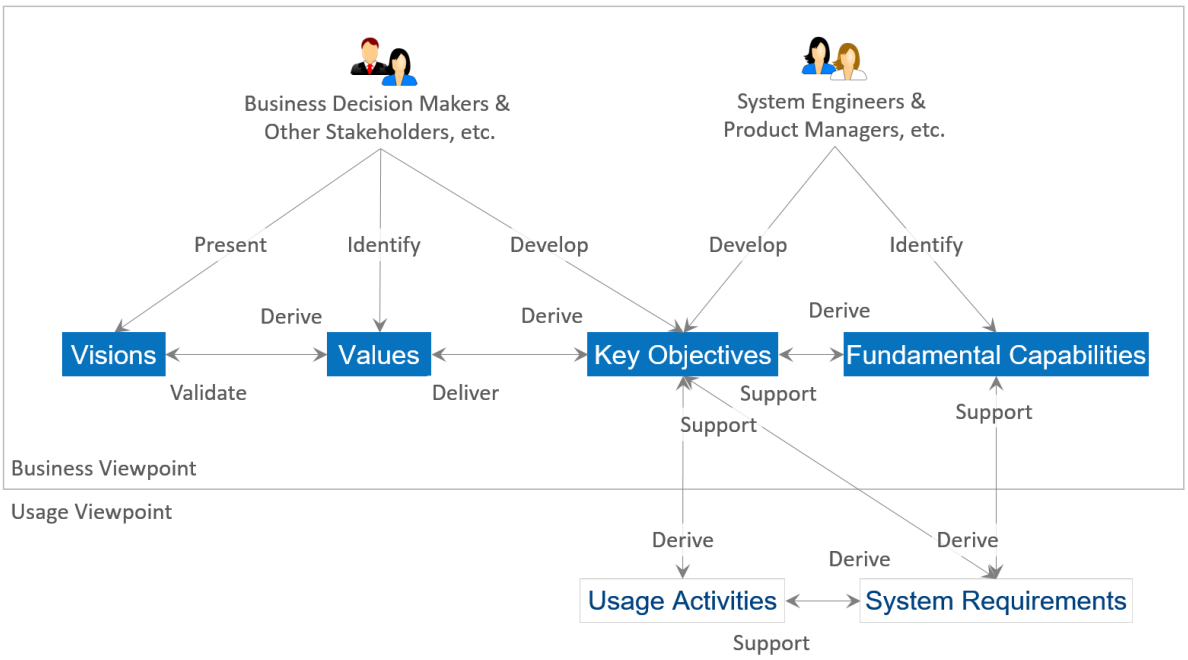


Figure 2.7-1: A Vision and Value-Driven Model

Stakeholders have a major stake in the business and strong influence in its direction. They include those who drive the conception and development of IIoT systems in an organization. They are often recognized as important strategic thinkers and visionaries within a company or an industry. It is important to identify these major stakeholders and engage them early.

In conceptualizing and defining an IIoT system, business stakeholders may take many technological and business factors into consideration, including technological trends, market conditions and potential, customer inputs and regulatory requirements (for example, safety, privacy, environmental and labor).

Vision describes a future state of an organization or an industry.[1] It provides the business direction towards which an organization executes. Senior business stakeholders usually develop and present an organization's vision.

Values reflect how the vision may be perceived by the stakeholders who will be involved in funding the implementation of the new system and by the users of the resulting system. These values are typically identified by senior business and technical leaders in an organization. They provide the rationale as to why the vision has merit.

Key objectives are quantifiable high-level technical and ultimately business outcomes expected of the resultant system in the context of delivering the values. Key objectives relate to the IIoT

[1] The concepts of vision, values and experiences and key objectives are related to the BMM concept of Ends (i.e., the results, or what needs to be achieved) [OMG-BMM].

system of interest and must involve the physical world. Key objectives should be measurable and time bound. Senior business and technical leaders develop the key objectives.

Fundamental capabilities refer to high-level specifications of the essential ability of the IIoT system to complete specific major business tasks.[1] Key objectives are the basis for identifying the fundamental capabilities. Capabilities should be specified independently of how they are to be implemented (neutral to both the architecture and technology choices) so that system designers and implementers are not unduly constrained at this stage.

Stakeholders first identify the vision of the organization and then how it could improve its operations through the adoption of an IIoT system. From the vision, the stakeholders establish the values and experiences of the IIoT system under consideration and develop a set of key objectives that will drive the implementation of the vision. From the objectives, the stakeholders derive the fundamental capabilities that are required for the system.

To verify that the resultant system indeed provides the desired capabilities meeting the objectives, they should be characterized by detailed quantifiable attributes such as the degree of safety, security and resilience, benchmarks to measure the success of the system, and the criteria by which the claimed system characteristics can be supported by appropriate evidence.

3.1 BUSINESS ASPECTS AND CONCERNS OF BUSINESS STAKEHOLDERS

The most common business viewpoint concerns include the following.

Business vision and value of the system:

- Overall vision and values, goals and stakeholders, expected ROI and business improvement.
- Costs—benefits covering the solution lifecycle: development, operation, evolution and maintenance). These include system evaluation criteria and what are their priorities.
- Requirements and Adaptation related to future business evolution, projections.

Contracts:

- Agreements and contracts internal to the organization: contributions and duties.
- Agreements and contracts external to the organization: partnerships, suppliers, outsourcing (services, subsystems).
- Service level and quality agreements (with users, customers, contributors).
- Evaluation process and quality targets: metrics, targets and penalties.

[1] A capability is normally defined as “the ability to do something” although in enterprise architecture terms it is extended to be “a high-level specification of the enterprise’s ability”. Fundamental Capabilities map to the Means aspect of the BMM, being a starting point for considering how the solution will provide the “means” to deliver the vision.

Regulations and compliance:

- Compliance for national and international markets.
- Industry-specific standards and policies.
- Trustworthiness goals, priorities and policies: definition of trustworthiness for this system, trade-offs regarding the desirable targets.

Governance:

- Development and deployment: roadmap and procedures.
- Usage policies: definition of responsibilities.
- System operation: management, responsibilities, costs.
- Procedures and protocols for ongoing evaluation, testing and maintenance.

4 USAGE VIEW

4.1 A DEFINITION OF USER AND USAGE

The *usage view* addresses the concerns related to the use of all or parts of an IIoT system.

The usage view relates to the operational aspect of using an IIoT system. In this operational context, the user is defined as an entity that intentionally interacts with an IoT system. Humans or other entities who benefit from the utilization of an IoT system without directly operating or interacting with the system are considered stakeholders and not users. When such stakeholders have a business interest, their concerns are addressed under the business viewpoint. Such business stakeholders may also be users if they directly interact with the system.

The user is by definition external to the IIoT system of interest. This is intended to distinguish the usage relationship clearly from interactions that are internal to a system.[1] The usage viewpoint is about concerns of entities external to a system. The user-system relationship is affected by a set of non-functional concerns from users (such as safety, security, ergonomics, operation costs and skills, regulations, performance or reliability), not just by functional aspects of the interactions (protocols, interfaces, operations).

A user may be a digital system, which uses the IIoT system of interest for the benefit of certain stakeholders, for example:

- an operation management system querying an IIoT system to obtain current operational status of the machines it monitors,
- an automated, non-human agent that exercises control on a system to contribute to the automation of decisions, such as AI, designed to convey an intention from human beneficiaries and

[1] It is possible to define a larger IoT system that includes some “user” entities for its sub-systems, but then we would not distinguish this internal usage from any other form of usage.

- another system with which the IIoT system is integrated, both systems acting as a sub-system in a larger system that serves broader objectives.

A user may be a digital system, which uses the IIoT system of interest for the benefit of certain stakeholders, e.g. an operation management system querying an IIoT system to obtain current operational status of the machines it monitors. In this case the external digital system may be considered a digital user. The digital user interacts with the IIoT system through an application interface running over a network interface. The following may be digital users:

- an automated, non-human agent that exercises control on a system. The agent typically involves technologies that contribute to the automation of decisions, such as AI, designed to convey an intention from human beneficiaries.
- another system with which the IIoT system of interest is integrated, both systems acting as a sub-system in a larger system that serves broader objectives. The largest system acts as a user of the IIoT system, toward serving the purpose (intent) it has been designed for.

In contrast to an IIoT system user, a physical entity-of -interest is an external entity that interacts with the IIoT system as a subject of the system by being monitored or controlled. Examples of physical entities-of -interest include a machine being monitored by a manufacturing IIoT system and a patient being monitored by a medical IIoT.

The boundaries of an IIoT system are a system-design decision. A system designer may arbitrarily define the scope of an IIoT system, provided that its components satisfy the system condition. In the previous example, a connected machine may be considered as a physical entity-of-interest (i.e. external to the system). In this case, it is connected via IoT devices of the system, subject to their sensing and actuating capabilities.

User interaction is needed to operate the system or to enable other users to operate the system to address stakeholders concerns and interests.

IIoT systems are often used to monitor and analyze biometrics, human language, senses, movement, and other behaviors. In these IIoT systems, the human being monitored and analyzed is not a user, but a subject being acted upon by the IIoT system, in the same way a machine is monitored by a sensor. In these cases, the human and the machine are considered as entities of interest of the IIoT systems.


Example

A healthcare worker who uses a connected sensor on a patient to monitor the patient's vitality (active with intention) is a user of the system of which the sensor is a part. The patient, though they interact with the sensor and therefore the system, is not a user because he or she is a (passive) subject of being "acted upon" by the system, similar to a sensor being attached to a machine. In this case the patient is taking on the role of the physical entity of interest.

> An IIoT system may include a work safety management subsystem that monitors workers in a work environment. Some of the workers may be a user of the IIoT system operating it for production. At the same time, the same workers may be monitored by the work safety sub-system without any action of their own. In that case, they are the subject or the physical entity-of-interest of the system, not a user, therefore, the workers take on both the roles of system user and physical entity of interest.

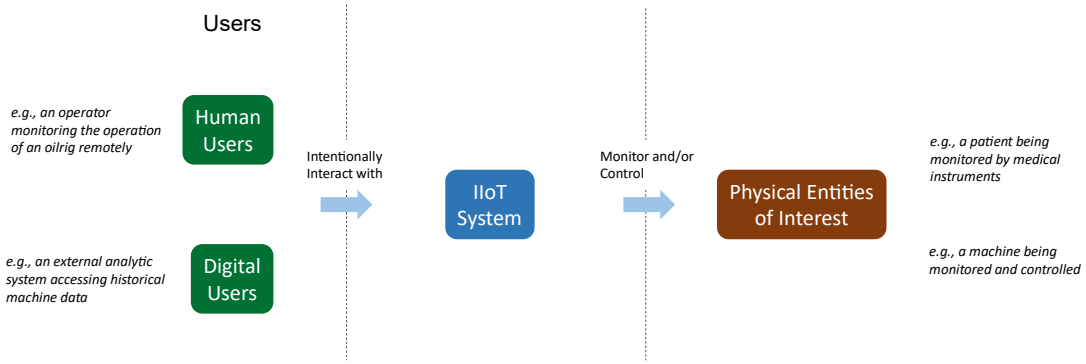


Figure 4.1-1: Outline of IIoT System Users and Entities of Interest

4.2 THE ASPECTS OF USAGE AND CONCERNS OF USERS

Using an IIoT system has costs and benefits. These benefits and costs may be measured and controlled with metrics that are often specific to particular industrial operations associated with the IoT system.

They include:

Efficacy and efficiency of operations: The functional adequacy and performance of operations include system performance, scalability and stability (benefits) and costs including financial, opportunity, time, equipment and other resources needed to maintain the operation of the system.

Usability and ergonomics, interface quality: Improving the ease of use and speed of task completion, reducing the costs of human risks such as musculoskeletal disorders, stress, injuries, skills and physical ability needed to use the system.

Evolution and adjustment of the system to the physical context, such as configuration, calibration and change management. Positive indicators include the flexibility of the system in accommodating a broader set of situations or tasks, and in evolving over time to accommodate new components and technologies. Costs include reducing the time and skills needed to change configurations.

System lifecycle management covers IT and OT lifecycles, maintaining service agreements, how easily the system can be tested before production to respond to various upgrades or contextual changes and the cost of reducing the time and skills needed for testing.

Required level of skills, training needs and human logistics: how easily can training material be produced, modified and assimilated by users; costs include reducing learning time and level of skills required.

Operational aspects of trustworthiness: improvement and costs involved in implementing and supporting system trustworthiness characteristics (security, safety, reliability, resilience, privacy), especially related to usage. These include the implementation and evaluation of safety, the rules for security and privacy and how easily they can be followed by users, the role of operators in ensuring the reliability and the resilience of system operations. Cost concerns include containing additional expenses, complexity, and inevitable trade-offs.)

5 FUNCTIONAL VIEW

5.1 BACKGROUND

Industrial control systems (ICSs) have been widely deployed to enable industrial automation across industrial sectors.[1] As we bring these automated control systems online with broader systems in the industrial internet, control remains a central and essential concept of industrial systems. Control, in this context, is the process of automatically exercising effects on physical systems and the environment, based on sensory inputs to achieve business objectives. Many control systems today apply real time, low-latency, fine-grained closed-loop controls to physical systems in close proximity to the machines they control, without a connection to other systems. Because of this, it is difficult to enable local collaborative control among machines in production processes, let alone larger scale orchestrated operations across production processes and even across manufacturing enterprises.

Some argue that the industrial internet is the conjoining of what has been traditionally two different domains with different purposes, standards and supporting disciplines: IT and OT.[2,3] In IT (information technology), everything is reducible to bits that represent ideas in the programmer's head and transformed in a way to produce useful inference—anything from the sum of numbers in a column to email systems to schedule optimization problems (e.g. using

[1] ICSs typically include programmable Logic Controllers (PLC), supervisory control and data acquisition (SCADA) systems [NCS-SCADA], distributed control systems (DCS) [Lydon-2011], and other control system configurations that are often found in the industrial control sectors [Petruz-2016].

[2] Here we use the more traditional version of the word 'domain'.

[3] Consider this an introduction to the topic—we will deal with the IT/OT problem in more detail in future versions of this and other documents.

Simplex). The essential problem with such an approach, noted as one of the fundamental problems in the artificial intelligence community, is the so-called 'symbol-grounding problem'—that symbols in the machine (the numbers passed around by the processor) only correspond to real-world objects because of the intentions of the programmer—they have no meaning to the machine.[1,2] In OT (operational technology), 'controls' have been applied directly to physical processes without any attempt to create symbols or models to be processed by the machine. For example, proportional-integrative-derivative (PID) controllers may control the voltage on a line using a particular feedback equation that is defined by the control engineer and demonstrated to work for a particular application—there is no attempt at generality.

The incidence of IT into the OT world has primarily come about due to a need to network larger systems and establish control over hierarchies of machines while also wanting to inject common IT ideas into the OT world (such as scheduling and optimization of resource consumption). There has also been a move toward controls that simulate the physical world digitally and base their control decisions on the simulation model rather than a control engineer's equation. This makes other kinds of approaches that have been examined in IT, such as machine learning, possible to apply to OT. This has also led to OT systems becoming susceptible to IT problems, such as network denial-of-service attack and spoofing and the symbol-grounding problem. The combination of IT and OT is a great advance—even opening a potential for cognition embodied into an industrial system creating possibility for innovation. This could avoid the symbol-grounding problem by basing its representation on the world (not on programmer-supplied models) and only its own episodic experience (and thus not limited to human conceptions of epistemology). However, even nearer-term breakthroughs that support advanced analytics based on real-world data rather than engineering models may yield substantial improvements. The main obstacles are safety and resiliency. Mission-critical OT applications, where failures would jeopardize life or human wellbeing, require reliability at a level much higher than those typically found in IT applications. Therefore, the reliability level typically found in IT systems may not be acceptable in mission-critical OT applications. Moreover, actions in the physical world generally cannot be undone, which is a consideration that IT systems normally do not have to address.

Advancement of computation and communication technologies of recent years can be applied to the industrial internet to transform industrial control systems in two major themes:

Increasing local collaborative autonomy: new sensing technologies provide more accurate data about the physical world. Greater embedded computational power enables more advanced analytics of these data and better models of the state of physical systems and the environment in which it operates. The result of this combination can lead the way of transforming control

[1] *http://en.wikipedia.org/wiki/Symbol_grounding_problem*; also see *The Symbol Grounding Problem* [Harnad-1990].

[2] *http://en.wikipedia.org/wiki/Chinese_room*; also see *Minds, Brains and Programs* [Searle-1980].

systems from merely automatic to autonomous, allowing them to react appropriately even when the system's designers did not anticipate all the possible system states. Ubiquitous connectivity between peer systems enables a level of collaboration that was previously impractical.

Increasing system optimization through global orchestration: Collecting sensor data from across the control systems and applying analytics, including models developed through machine learning in addition to physical or engineering models, to these data, can provide insight to operations of the physical systems. With these insights, improved decision-making and optimized system operations can be achieved globally through automatic and autonomous orchestration.

These two themes have far-reaching impact on the systems that we will build, though each system will have a different focus and will balance the two themes differently.

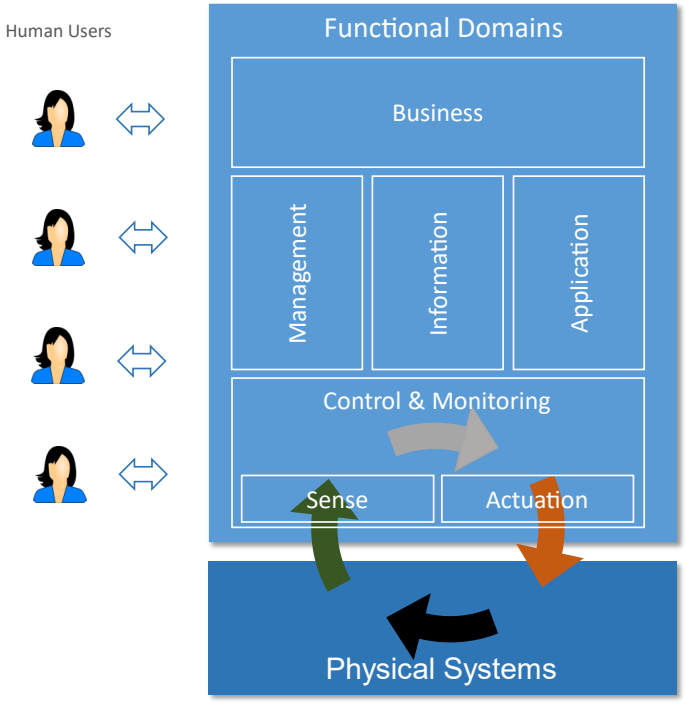
5.2 THE FUNCTIONAL VIEW

The *functional view* addresses the concerns related to the functional capabilities and structure of an IIoT system and its components.

A *functional domain* is a (mostly) distinct functionality in the overall IIoT system. A decomposition of a typical IIoT system into functional domains highlights the important building blocks that have wide applicability in many industrial verticals. It is a starting point for conceptualizing a concrete functional architecture. Specific system requirements will influence how the functional domains are decomposed, what additional functions may be added or left out and what functions may be combined and further decomposed.

We decompose a typical IIoT system into five functional domains:

- Control & Monitoring Domain
- Information Domain
- Application Domain
- Business Domain
- System Management Domain



Green Arrows: Data/Information Flows; Grey/White Arrows: Decision Flows; Red Arrows: Command/Request Flows

Figure 5.2-1: Functional Domains

Data flows and control flows take place in and between these functional domains. Figure 5.2-1 above illustrates how the functional domains relate to each other with regard to data and control flows. Green arrows show how data flows circulate across domains. Red arrows show how control flows circulate across domains. Other horizontal arrows illustrate some processing taking place within each domain, to process input flows and generate new forms of data or control flows.

Controls, coordination and orchestration exercised from each of the functional domains have different granularities and run on different temporal cycles. As it moves up in the functional domains, the coarseness of the interactions increases, their cycle becomes longer, and the scope of impact likely becomes larger. Correspondingly, as the information moves up in the functional domains, the scope of the information becomes broader and richer, new information can be derived, and new intelligence may emerge in the larger contexts.

5.3 The Control And Monitoring Domain

The *control and monitoring domain* is a functional domain is the collection of functions performed by industrial control and automation systems, and other physical systems outside of conventional industrial sectors, for example, medical devices, heating, ventilation, air conditioning (HVAC) systems, and traffic control systems. The core of these functions comprises fine-grained closed-loops, reading data from sensors (“sense” in the figure), applying rules and

logic, and exercising control over the physical system through actuators ("actuation").[1] Such closed-loop systems usually require high timing accuracy and resolution. They are typically implemented using proven control technologies such as PLCs. Components or systems implementing these functions (functional components) are usually deployed in proximity to the physical systems they control and may therefore be geographically distributed. These components and systems may not be easily accessible physically by maintenance personnel, and physical security of these systems may require special consideration. On the other hand, there are increasing number of industrial automation systems that are mobile, e.g. robotic machines in a manufacturing floor; providing low latency and reliable wireless communication to and between these mobile systems requires additional consideration.


Example

Simple examples of functional components in this domain include control units in electricity utility plant, control units in a wind-turbine, and control units in autonomous vehicles.

The control domain comprises a set of common functions, as depicted in Figure 5.3-1.[2] Their implementation may be at various levels of complexity and sophistication depending on the systems, and some components may not exist at all. These are abstract functions that many control or automation units provide in various forms and in different architectures.

An IIoT system seeks to establish connectivity, to gather data and provide optimization feedback through advanced analytics, often in a larger context than the individual control and automation system. It is not intended to replace or alter the existing implementation of control and automation system in the control domains. Some IIoT systems may monitor physical systems, providing visibility of their operational state, including alerts on operational exceptions, but do not issue instructions back to the physical systems, relying on human operators to intervene should it be necessary (human-in-the-loop). Others IIoT systems may seek to provide high level feedback to the control systems, by providing high-level control targets (such as setpoints to a PID controller), rather than real-time fine-grained closed-loop control.

We describe each below.

Sensing is the function that reads sensor data from sensors. Its implementation spans hardware, firmware, device drivers and software elements.

Actuation is the function that writes data and control signals to an actuator to enact the actuation. Its implementation spans hardware, firmware, device drivers and software elements.

[1] Possibly in a hierarchy, at several levels.

[2] These functions are considered essential to many control systems in the control domain. However, they may exist in other domains as well.

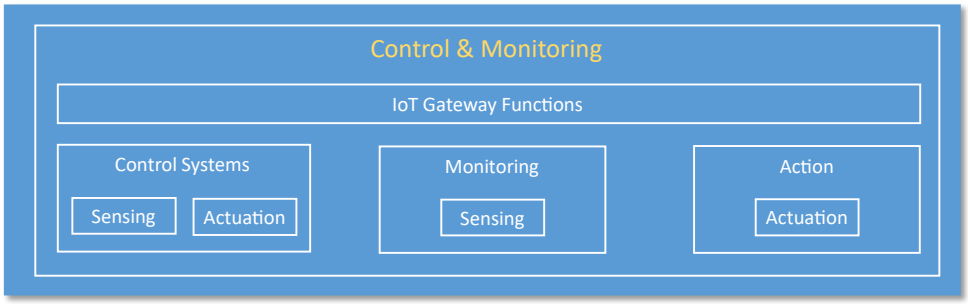


Figure 5.3-1: Functional Decomposition of Control Domain

Control is the function in which closed-loop feedbacks are implemented that includes sensing (reading data from its sensors), performing control computation and actuation (outputting signals to the actuators). The control calculation involves understanding the states, conditions and behaviors of the systems under control and those of peer systems by interpreting and correlating data gathered from sensors and peer systems. The complexity and sophistication of control computation of the system under control varies greatly. It may range from straightforward computation (such as a simple interpretation of a time series of the temperature of a boiler), to moderately complex (a prebuilt physical model of an aircraft engine), to very complex and elastic (models built with artificial intelligence possessing learning and cognitive capabilities). These computation capabilities, sometime referred to as edge analytics, are generally required to be evaluated locally in control systems for real-time applications. Edge analytics are also needed in use cases where it is not economical or practical to send a large amounts of raw sensor data to remote systems to be analyzed even without a real-time requirement.

Monitoring is the function in which only sensing is performed (e.g. temperature and wind speed measurements in a weather station).

Actuation is the function in which only actuation is performed (e.g. a controller that accepts commands from other systems to open or close a gate).

Although a new generation of sensors, actuators and automation control systems are capable of communicating with an IIoT system using 'modern' internet protocols, e.g. PLCs that support the OPC UA protocols [OPC-UA], there are vast number of established industrial controllers or other physical systems are still using traditional industrial protocols. To connect to these established (or legacy) systems, IoT gateways can be deployed to bridge the communication gap between the legacy systems and IIoT systems.

An IoT gateway provides a set of functions that bridge the gap in communication between legacy industrial systems and IIoT systems. For example, a bridge function may include communication protocol translation (e.g. from [MODBUS] to [MQTT]), communication mode adaptation (e.g. acting as a client to both an [OPC-DA] server and IIoT system data gathering server), data format (syntactic) translation (e.g. from binary readings to JSON format), data meaning (semantic)

translation (e.g. performing unit conversion). It may include some edge computing functions such as fast Fourier transformation (FFT) on vibration data and deep learning-based video analytics.

5.4 The System Management Domain

IIoT systems tend to be complex systems, including many functional components that are loosely coupled and distributed. The *system management domain* manages them.


Example

Optimizing the operation of one train has obvious cost savings but optimizing train operations and routes across a fleet yields more and combining data from fleets owned by different railroads can optimize the utilization of the rail network within a country.

Figure 5.4-1 shows the system management domain with its functional decomposition.

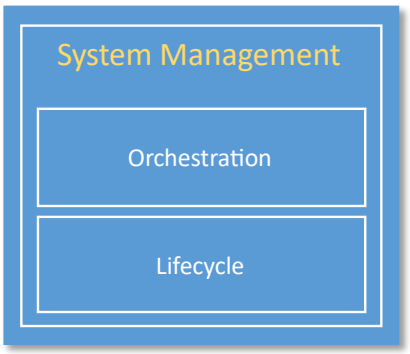


Figure 5.4-1: System Management Domain decomposition showing support across various customers

Lifecycle comprises conventional software-system full-cycle management functions required to deploy, configure, monitor, diagnose and update the IIoT system and its sub-systems.

Orchestration comprises functions required to coordinate the operation of the various subcomponents of an IIoT system that tend to be loosely coupled and distributed and the interactions with outside systems.

5.5 The Information Domain

The *information domain* is the collection of functions for gathering data from various domains, most significantly from the control domain, and transforming, persisting, and modeling or analyzing those data to acquire high-level intelligence about the overall system.[1] The data collection and analysis functions in this domain are complementary to those implemented in the control domain. In the control domain, these functions participate directly in the immediate control of the physical systems whereas in the information domain they are for aiding decision-making, optimization of system-wide operations and improving the system models over the long

[1] Possibly in a hierarchy, at several levels.

term. Components implementing these functions may or may not be co-located with their counterparts in the control domain. They may be deployed in building closets, in factory control rooms, in corporate datacenters, or in the cloud as a service.

Example

Optimizing the electricity generation level of a plant or a generator based on the condition of the facility, fuel cost and electricity price.

Changing the route of a fleet of freight trucks based on weather, traffic and the condition of the goods in the trucks.

Changing the output of an automated production plant based on condition of the facility, energy and material cost, demand patterns and logistic.

Changing the temperature set-point of a boiler based on energy cost, weather condition and usage pattern.

Figure 5.5-1 illustrates the functional decomposition of the information, application, and business domains.

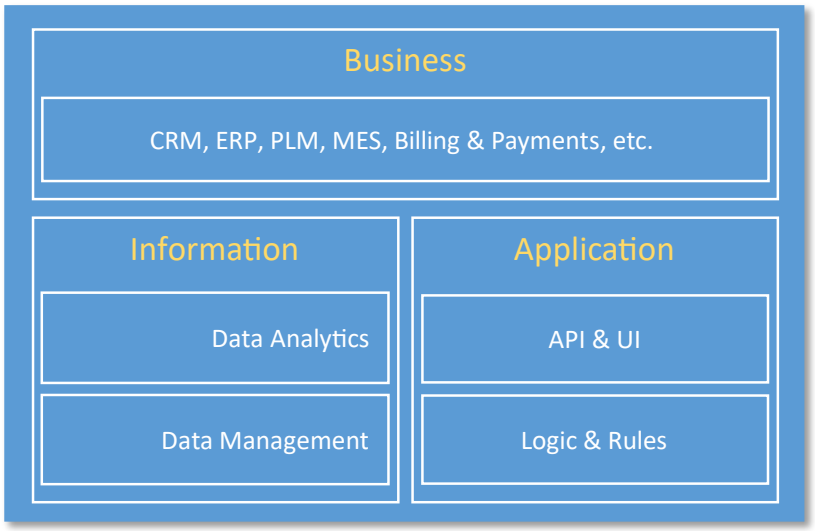


Figure 5.5-1: Functional Decomposition of Information, Application & Business Domains

Data Management[1] consists of functions for:

- ingesting sensor and operation state data from all domains,

[1] The data management functions described here cover the functions concerning data gathered from sensors and other systems so they can be used for data analytics effectively. Many of these functions may be implemented distributed across various functional domains and components, making distributed data management a crosscutting concern. Nevertheless, the data management functions are most likely to considered in the information domain, accounting for what functions are provided by other domains and remains to be implemented before data analytics.

- quality-of-data processing (data cleansing, filtering, de-duplication, etc.),
- syntactical transformation (e.g. format and value normalization),
- semantic transformation (semantic assignment, context injection and other data augmentation processing based on metadata (e.g. provisioning data from the Operations Domain) and other collaborating data set,
- data persistence and storage (e.g. for batch analysis) and
- data distribution (e.g. for streaming analytic processing).

These functions can be used in online streaming mode in which the data are processed as they are received to enable quasi-real-time analytics in support of orchestration of the activities of the assets in the control domain. They may be used in offline batch mode (e.g. seismic sensor data collected and accumulated in an offshore oil platform that does not have high-bandwidth connectivity to the onshore data center).

Data governance functions may be included for data security, data access control and data-rights management, and conventional data management functions related to data resilience (replication in storage, snapshotting and restore, backup & recovery and so on).

Data analytics encapsulates a set of functions for data modeling, analytics and other advanced data processing, such as rule engines. The analytic functions may be done in online/streaming or offline/batch modes. In the streaming mode, events and alerts may be generated and fed into functions in the application domains. In the batch mode, the outcome of analysis may be provided to the business domain for planning or persisted as information for other applications.

The data volume at the system level in most IIoT systems will eventually exceed a threshold at which the traditional analytic toolsets and approaches may no longer scale in meeting the requirement in performance. “Big data” storage and analytic platforms may be used to implement these functions.

Scalability and flexibility are indispensable in an IIoT system that performs various analytic processing on data. Data management and analytics, although loosely coupled, are separate functions that can be implemented and optimized independently, allowing higher scalability than if they were closely-coupled.

5.6 THE APPLICATION DOMAIN

The *application domain* is a functional domain for implementing application logic. It comprises the functions that implement application logic that realize specific business functionalities. Functions in this domain apply application logic, rules and models at a coarse-grained, high level for optimization in a global scope. They do not maintain low-level continuing operations, as these are delegated to functions in the control domain that must maintain local rules and models in the event of connectivity loss. Requests to the control domain from the application domain are advisory so as not to violate safety, security, or other operational constraints.

The decomposition of the application domain is illustrated in Figure 5.5-1.

Logics and rules comprise logic (rules, models, engines, activity flows, etc.) implementing specific functionality that is required for the use case under consideration. There are great variations in these functions in both its contents and its constructs among the use cases.

APIs and UI are expose functionalities as APIs for other applications to consume, or human user interfaces enabling human interactions with the application.

5.7 THE BUSINESS DOMAIN

The *business domain* implements business logic and interacts with backend systems. It supports business processes and procedural activities that an IIoT system must integrate to enable end-to-end operations of IIoT systems (i.e. providing adapters for connectivity with backend systems). Examples of these business functions include Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), Product Lifecycle Management (PLM), Manufacturing Execution System (MES), Human Resource Management (HRM), asset management, service lifecycle management, billing and payment, work planning and scheduling systems.


Example

A predictive maintenance service for an oil rig may have an application that forecasts failures in the field. To do so, it may require a resource planning system to ensure the required parts are available and reserved, and it may need to connect to internal or partner's service work schedule system and logistics management system, and the customer's, to schedule the field service.

5.8 CROSSCUTTING FUNCTIONS AND SYSTEM CHARACTERISTICS

The functional domain's components focus on major system functions that are required to support generic IIoT usages and to realize generic IIoT system capabilities for business purposes. However, additional functions must be provided to enable the major system functions. These crosscutting functions need to be made available across many of the system functional components. For example, system functions need to be connected so they can interact with each other to complete functionality at the system level. Other crosscutting concerns are system management, to assure a certain level of system availability and scalability and distributed data management for analytics on the data gathered from the industrial assets and control systems to gain insights on their operations.

On the other hand, the aggregate behavior of an IIoT system is not the sum of what is provided by its constituent functional components. Like any complex system, there are emergent behaviors or properties resulting from the interactions of the constituent parts. These emergent system-wide properties are called system characteristics. See section 2.6.

The system and crosscutting functional analysis concern how the system works while the analysis of system characteristics emphasizes how well the system works. For example, to ensure security

in a system, a certain set of security functions must be implemented in each of the functional components and their communications, such as encryption and authentication. How secure the system-as-a-whole is depends on how these functions are implemented and how securely these functional components are integrated and interact with each other—as an emergent property. A system is as secure as its weakest component, or link between components. The same is true for safety, resilience and any other system property. These system characteristics (safety, security, resilience, reliability and privacy) is defined as *trustworthiness*. Refer to the *Industrial IoT Trustworthiness Framework Foundations* [IITF] for detailed analysis of the trustworthiness of IIoT systems.

The realization of a system characteristic to a certain desired level may depend on, constrain or conflict with other system characteristics. For example, one cannot ascertain that a system is safe without also ascertaining it is secure. On the other hand, an inadequately implemented security measure may be hazardous to safety.[1] Refer to *Industrial IoT Trustworthiness Framework Foundations* [IITF] for detailed analysis on the entwined dependencies and conflicts that can occur among and between system characteristics.

This reference architecture places a strong emphasis on both the functions needed to support the system's business purpose and ensuring adequate system characteristics so that the functions are performed correctly and the business purpose is not compromised. The crosscutting functions and system characteristics are discussed in *Key IIoT System Concerns* [IIC-KSC], *Industrial Internet Security Framework* [IIC-IISF], *Industrial IoT Trustworthiness Framework Foundations* [IITF], and elsewhere.

The relationship between the functional domains, crosscutting functions and key system characteristics are summarized in Figure 5.8-1:

[1] For example, a locked-up fire escape door may provide strong security against unauthorized entrance or exit but may also prevent necessary escape in an emergency.

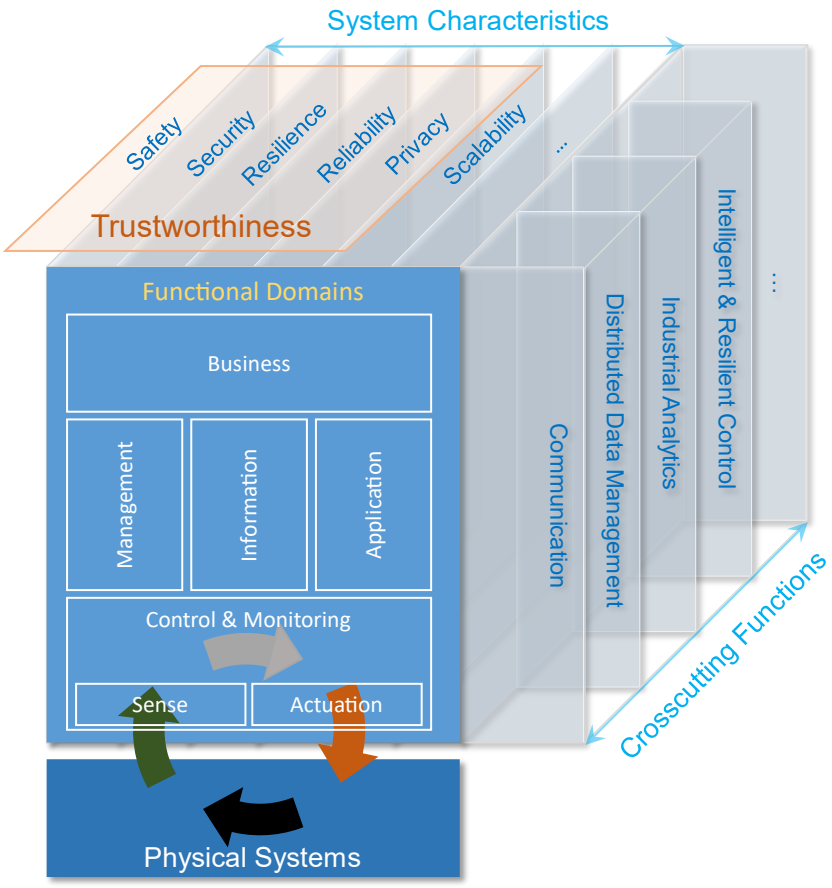


Figure 5.8-1: Functional Domains, Crosscutting Functions and System Characteristics

5.9 Functional Domains and Computational Deployment Patterns

The convergence of operational technology and information technology that enables the industrial internet is driven by technology advances in ubiquitous connectivity and pervasive computation. Mapping the IIoT functional domains to connectivity and computational deployment patterns is a high-level guide to how these functional domains could be distributed.

Connectivity is the foundation connecting the computational capabilities, enabling information sharing and collaborative operations among computers, machines and people. Near the network peripheries, advances in connectivity, such as high-performance and low-power wireless communication, make it possible to connect to large numbers of industrial assets without the cost of laying wires to reach them. Within large data centers, Software Defined Network (SDN) is maturing, making it possible for applications to manage their networking dynamically.

Meanwhile, technologies concerning computational deployment patterns, which involve the location and placement of computational capability (including applications, data and services), continue to evolve.

On one hand, large-scale computation capability has become available on demand with unprecedented scalability, accessibility, availability and elasticity at low cost through the economy of scale at large data centers, thereby leading to the advent of cloud computing. This is made possible by advances in virtualization technologies, including containerization technologies, and the maturing of infrastructure-as-as-service (IaaS), platform-as-a-service (PaaS), software-as-a-service (SaaS) technologies.

The cloud computing approach and platforms in virtualizing and managing computation resources are being more broadly adopted in enterprise data centers. These computational patterns characterized by a high concentration of computing capabilities in public-, private-, hybrid-clouds, on premise and remotely hosed data centers—collectively called the *concentrated computational pattern*, or the concentrated pattern for short—offer unparalleled scale of computational resources with elasticity at low cost.

On the other hand, computational capabilities ranging from limited (e.g. in a smart sensor) to rich (e.g. a cluster of servers) are placed in, attached to or collocated with physical systems (e.g. in smart sensors, devices or machines). These capabilities are connected through a variety of communication technologies including wireless. Some of the computational capabilities and communication technologies are energy-constrained (e.g. powered by a battery). This computational deployment pattern is called the *dispersed computational pattern*, or dispersed pattern for short.

Traditional industrial control systems are examples of dispersed computational patterns at the network peripheries where computation, albeit mostly embedded, is performed at controllers connected to a network or in isolation.

There is a renewed movement to distribute computational capability toward the network peripheries, away from concentrated computation centers, to benefit from a reduction in movement of data and communication latency, and enhancement in local intelligent control and, more importantly, resilience [IIC-vPLC].

Moreover, some of the technologies that have been originally developed for large data centers, such as virtualization and SDN, are being applied to manage computational and network resources in dispersed patterns at the network peripheries as well.[1]

[1] The term ‘cloud’, as in cloud computing, is often used loosely, its exact meaning depending on the context. Commonly, it refers to computing resources located somewhere “out there in the cloud” as in some remote data centers. Cloud computing involves approaches and technologies for managing computing resources that are being extended to be used outside of data centers, such as those deployed near physical systems in what is now commonly referred to as “edge computing”. Refer to IIC “Distributed Computing in the Edge” [IIC-DCE] for more such details.

Generally, it is an architectural choice which computational patterns to use and where to place various functions needed for an IIoT system. With advances in technologies available at the network peripheries, including the availability of smarter sensors and devices, most IIoT systems will involve the dispersed computational deployment pattern in conjunction with a concentrated pattern. Employing the concentrated pattern provides a higher degree of elasticity of computational capability and a simpler management of the computational resources. The concentrated pattern can be deployed anywhere across a network, including near or at its peripheries to benefit from a reduction in network bandwidth and latency, stronger local control and resilience. In many use cases, functions needing high computational capabilities with less stringent latency and reliability requirements can be placed in the concentrated pattern away from the network peripheries. Employing the dispersed pattern at or near the network periphery exclusively, e.g. through a pure peer-to-peer collaborative model, yields the greatest resilience.

Some IIoT architectures adapt the concentrated pattern with a relatively flat and thin layer at the network peripheries leaving most of the computation being performed away from the periphery.


Example

For example, it may be efficient to connect a large number of remote sensors that measure air quality and other environmental parameters in a large metropolitan area to a cloud service and perform most of the analytics computation therein.

This computational pattern may not be adequate for many IIoT systems where computation needs to be distributed across multiple, potentially hierarchical, layers close to the industrial assets at or near the network periphery. When industrial assets are dispersed and remote, e.g. turbine engines in a wind farm or oil rigs in an oil field, strong computational capability may be needed at or near the assets for local analytics and control.

As the industrial internet matures, more computation capability will be added to or placed adjacent to industrial systems to enable local intelligent and autonomous operations making computation more dispersed.

Traditionally, operational technology is deployed around the network periphery while information technology is deployed away from them. Functions in the control and operations domain map to the operational technology and functions in the business, information and application domains map to the informational technology. However, these boundaries are vague to begin with and will continue to blur as the industrial internet matures and the operational technology and information technology converges.

The IIRA does not constrain how its functional domains are distributed across the network or the computational patterns used. An optimal distribution pattern of the functional domains will be determined by the specific system context and requirements, the demand on the availability of the computational resource and the available technologies to support such a distribution pattern.

5.10 Functional Requirements Categories

A list of requirement categories and their functional descriptions helps to map logical functions with implementation patterns that are described in Chapter 6:

Category	Description
Control	Functions for implementing control logic
Physical Interaction	Functions for interacting with the physical world. Interacting functions can be broken down into two categories: sensing and actuating. Every IoT system must have at least one sensing and/or actuating function.
System management	Functions for managing the IoT components of an IoT system
Data management	Functions for managing data, including the creating, accessing, updating, persisting, using, assuring, and destroying of data.
Communication	Functions for transfer of information between internal system components.
Security/Trustworthiness	Functions for assuring the IoT system meets trustworthiness requirements, including encryption and access control functions.
Internal Interface	Functions for interacting with other components, including digital and human user interfaces.
External Interface	Functions for interfacing with external systems, including digital and human interfaces
Data Analysis	Functions for analyzing data.
Logic	Functions for decision making.

Table 5.10-1: Functional Requirements Categories

5.11 Human Roles in the Creation and Operation of an IIoT System

A person may play any number of roles in an IIoT system. This section focuses on the roles people play in operating an IIoT system from a functional point of view.

As described in **Error! Reference source not found.**, both human and system (or part of a system) play roles in carrying out tasks to complete an activity. These roles differ for each IIoT system. They may be abstracted in some way from the functional point of view, as shown in Figure 5.2-1.

The *human role with the control domain* is largely consistent with common practice in deploying, operating and maintaining industrial control and automation systems. With the connectivity brought to these systems, more of these practices can be done remotely and more effectively. Increased deployment of robotic machines in mixed work environments with human operators on the factory floor results in humans playing a more cooperative role with the machines.

The *human role in the system management domain* largely consists of monitoring and maintaining the IIoT systems to ensure its continuing and optimal operations.

The *human role in the information domain* mostly concerns data processing and management, as well as building, validating and deploying analytic models to gain insights from the data. The effectiveness of an IIoT system will increasingly depend on the quality of the analytic models.

The *human role in the application domain* deals with how to apply business logic to the analytic insights to optimize the operation of the machines, and how to use these application functions in daily operational processes.

The *human role in the business domain* from the industrial internet perspective is about how to use the insights gathered from the data to optimize business processes and decision making.

People can potentially take on three types of *roles during the operation* of a given IIoT system: users, entities of interest and participants. First, a person may be the user of the IIoT system, acting as an external entity that interacts with the IIoT system to gain some benefit. Second, a person may be an entity-of-interest that the IIoT system is monitoring and acting upon. Finally, people may participate as part of an IIoT system. In other words, people may interact with other components in the IIoT system to perform some function necessary for the success of the system operation. This final role is especially important due to the challenges of understanding—what capabilities a given person will provide, how those capabilities fit into the system design as a whole and assuring that person is actually providing those capabilities when needed.

Not all humans support the goals of the IIoT system. *Intentional and unintentional human actions* can lead to system failures, performance degradation, security problems, privacy breaches, etc. Well-designed IIoT systems should be robust in the face of human-generated problems; these can include design errors, other user errors, implementation defects, configuration mistakes, intrusions, hacking, physical attacks and sabotage. For more details for how to deal with these negative aspects of human roles with regard to an IIoT system, refer to the concepts of trustworthiness described in the *Industrial IoT Trustworthiness Framework Foundations* [IITF].

6 IMPLEMENTATION VIEW

The *implementation viewpoint* addresses the concerns related to implementing the capabilities and structure of an IIoT entity of interest system. It concerns the technical representation of an IIoT system, and the technologies and system components required to implement the activities and functions prescribed by the usage and functional viewpoints.

An IIoT system architecture and the choice of the technologies used for its implementation are also guided by the business viewpoint, including: cost and go-to-market time constraints, business strategy in respect to the targeted markets, relevant regulation and compliance requirements and planned evolution of technologies.[1] The implementation must also meet the system requirements including those identified as key system characteristics that are common across activities and must be enforced globally as end-to-end properties of the IIoT system.

This reference architecture focuses on system conceptualization and architecture highlighting important system concerns that may affect system-lifecycle processes that includes system design, development, deploy and operation. Each of these system- entity of interest lifecycle phases have a unique set of concerns. We do not cover concerns about system deployment or operation here, especially given the diverse deployment models in different environments across industrial verticals, as they were briefly discussed in section 5.9.

Since IIoT system architecture is so diverse, it cannot be described using only one model. Instead, multiple models, or patterns, are needs to address different architecture structures found in common IIoT systems.

6.1 ARCHITECTURE PATTERNS

Architecture patterns are common, typical or essential features of IIoT implementations that are easy to recognize and understand by practitioners. Architecture patterns represent the structure of a system or part of a system. They are examples and references for conceptualizing real world IIoT architectures that can be used to develop new architectures by leveraging knowledge about existing systems to build new systems and understand current systems with increased confidence. The IIoT architect may arrive at a final architecture that combines several patterns and may be substantially different from them.

IIoT system implementations typically follow well-established architectural patterns entity of interest such as:

- IoT Component Capability Model Pattern
- Three-tier architecture pattern
- Gateway-Mediated Edge Connectivity and Management architecture pattern
- Digital Twin Core as a Middleware Architecture Pattern
- Layered Databus pattern
- System of Systems Orchestrator Architecture Pattern

An architecture pattern is a simplified and abstracted view of a subset of an IIoT system implementation that is recurrent across many IIoT systems yet allows for variants. For example, an implementation of the three-tier pattern in an IIoT system does not exclude multiple

[1] This version of the IIRA does not attempt to address regulatory and compliance requirements. These are substantially different by vertical and may be addressed in more detail in future documents.

implementations of every tier—e.g. many instances of the edge tier—and many-to-many connections between instances of a tier and instances of the next tier. Each tier and its connections will still be represented only once in the pattern definition. The following lists some common patterns found in IoT systems. For the complete set visit IIC Patterns[1] repository.

6.2 IoT Component Capability Model Pattern

The set of capabilities of a given IoT component must be understood to use it with confidence. A definition of capability is "the quality of being able to perform a given function".

Many IoT components are black boxes, meaning the organizations acquiring, using and administering them have little or no access to information about their internal workings, including the capabilities they offer. For white-box IoT components, where detailed information about the internal workings is available, there are seldom standardized mechanisms employed to expose, access, and configure capabilities.

It is useful to have a standardized approach to describe IIoT components' capabilities. This model uses a black-box approach to focus on the functionality that an IIoT component can provide to a system. Functionality that is internal to an IoT component is not visible using this model.

Figure 6.2-1: provides a model of the capabilities of an IIoT component. The large grey box represents an IIoT component, and the remaining boxes represent capability types for it. Each component can then be characterized by the set of capabilities it provides. It may have more than one of any given capability type (e.g. sensors, network interfaces, actuators), but it must have at least one capability that is provided through a network interface (a second capability). This model can also be used to describe the IIoT components and relationships formed into an IIoT system, or it can be used to describe the capabilities of an IIoT system.

[1] The IIC Patterns repository can be found at *https://www.iiconsortium.org/patterns/*

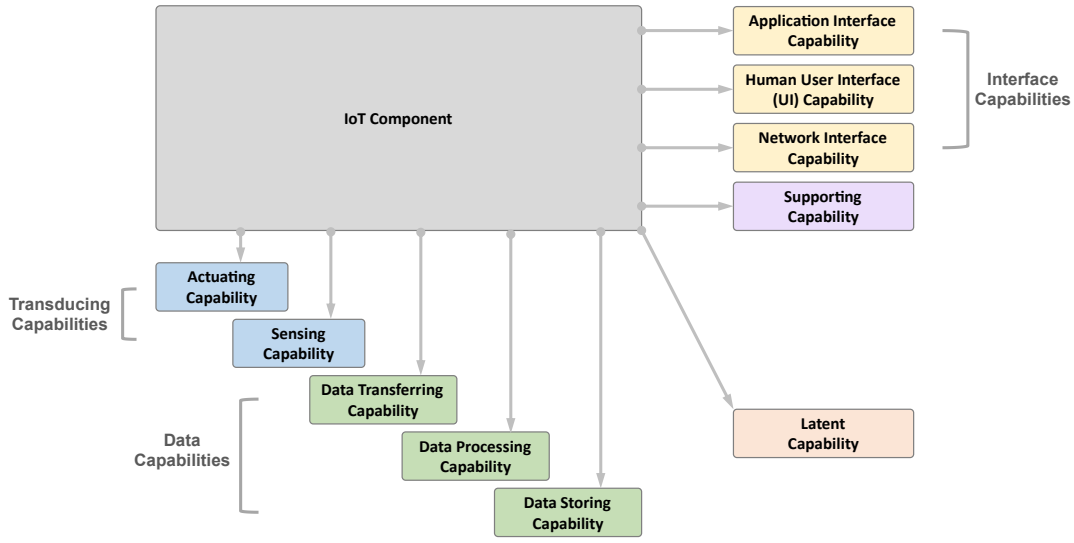


Figure 6.2-1: Capabilities of an IoT component

The IoT capabilities can be grouped into several categories:

- *Transducing capabilities* interact with the physical world. These capabilities, prevalent in OT, serve as the boundary (edge) between the digital and physical environments. Transducing capabilities let computing systems interact directly with physical entities of interest.
- *Data capabilities* are directly involved in providing functionality to the system. These capabilities—data storing, transferring, and processing—are commonly associated with conventional IT systems, and are critical to IoT systems.
- *Interface capabilities* provide the component the ability to interact with other IoT components (including people using a connected device to interact with the other system components).
- *Supporting capabilities* are indirectly involved in providing functionality to the system, such as monitoring, management, security, or orchestration.
- *Latent capabilities* are transducer, data, interface, or supporting capabilities that are not currently enabled and accessible outside the IoT component. These capabilities can potentially be enabled either by a trusted actor or a bad actor with malicious intent.

The subsections below provide additional information on each capability category.

6.2.1 TRANSDUCER CAPABILITIES

An *actuating capability*, provided by an *actuator*, offers the ability to make a change in the physical world based on information given as input to the component.

Errors may be introduced in the digital logic, the digital to analog converter, the analog electrical circuit, and the actuator transducer. There is a time delay between the input data arriving at the component and the change being made to the environment.

Examples of actuating capabilities include heating coils (heating capability), electric shock delivery (cardiac pacing), electronic door locks (lock/unlock capability), unmanned aerial vehicle operation (remote control), servo motors (motion capability) and robotic arms (complex motion capability).

An important type of actuator is a black-box control system that accepts a desired outcome as an input and internally uses sensors, actuators and processors to make the physical changes. This is considered an actuating capability in this model since the sensors and processors are not directly usable from outside the component.

A *sensing capability* (provided by a *sensor*) provides an observation of an aspect of the physical world in the form of measurement data. Information from sensor observations may be provided to other IoT components through the component's network interface for processing and storage.

Sensing is "read only"; any change to the physical state is a side effect. Measurement errors may be introduced by the physical environment between the physical system and the sensor transducer, in the sensor transducer itself, in the analog electrical circuit, in the analog to digital (A/D) converter and in the digital logic of the sensor. There is also a time delay between the sensing and the data becoming available at the component output.

Examples include temperature sensing (temperature measurement capability), computerized tomography (CT) scans (radiographic imaging), spatial sensing (accelerometers, gyroscopes), optical sensing and audio sensing.

6.2.2 DATA CAPABILITIES

A *data-storing capability* provides the ability to store and retrieve data and information over time. The intent is to store data for use at some later time. Data persists for a finite period. Data may be published by the component or provided automatically by design or in response to an external request. There is a time delay between the input and output, i.e. between a data request and the data response.

Examples of data-storing capabilities include databases, data brokers (such as a Message Queuing Telemetry Transport (MQTT) broker [Thomson-1985] and any other type of component that stores input data for later use.

A *data-transferring capability* provides the ability to transmit data from one physical or logical location to another. The data transferring capability provides the ability to 'black box' a network and provide information about the network without having to understand the specific topology.

As the interactions of an IIoT system with the physical world require the data transferring network to meet latency, reliability, and security requirements, it is useful to be able to describe the network characteristics in this manner, so the capability is explicitly called out by the IIoT general model.

Examples of data-transferring capabilities include data networks based on Ethernet, Institute of Electrical and Electronics Engineers 802.11 [IEEE-WiFi], and Long-Term Evolution [IIC-DCE] The Industrial Internet of Things Distributed Computing in the Edge, version 1.1, 2020

www.iiconsortium.org/pdf/IIoT-Distributed-Computing-in-the-Edge.pdf

[IIC-DTC] Digital Twin Core—Essential Elements for Interoperability, version 1.0, 2022

www.iiconsortium.org/foundational-publications/

[LTE-4G].

A *data-processing capability* provides the ability to transform data based on an algorithm. The intent of processing is to transform input data to provide output data. There is a time delay between the input and output that should be accounted for. The transformation may be simple, with a single input variable and a single output, or it may be complex with multiple inputs and outputs.

Control algorithms are an important type of data processing that take the output of sensor(s) and actuator(s) or pre-processor(s) and provide an output that can be fed into an actuator or post-processor. These control algorithms are often used within negative feedback loops. A proportional-integral-derivative (PID) control algorithm is an example of such a control algorithm.

Data analytics is another type of data processing. Data analytics encapsulates a set of functions for data modeling, analytics, and other advanced data processing, such as rule engines. The analytic functions may be done in online/streaming or offline/batch modes. In the streaming mode, events and alerts may be generated and fed into functions in the application domains. In batch mode, the outcome of analysis may be provided to the business domain for planning or persisted as information for other applications.

Other examples of processing include data aggregation, binary (true/false) analysis, big data analytics, machine learning, and predictive analysis.

6.2.3 INTERFACE CAPABILITIES

An *application-interface capability* provides the ability for other IIoT components (components, systems, etc.) to communicate with a given IIoT component through an IIoT component application. A widely used type of application interface is an application programming interface (API).

A *human-user interface (UI) capability* provides the ability for the component to communicate directly with people. Not all IoT components have a human UI capability (i.e. a dedicated processing component). Any effect on the physical environment is a side effect of the interface (the purpose being information exchange) and so is not considered to be sensing or actuating. Examples of human UI capabilities include keyboards, mice, microphones, cameras, scanners, monitors, touch screens, touch pads, speakers and haptic devices.

A *network-interface capability* provides the ability to interface with a digital communication network to communicate data from one component to another. Every IIoT component must have at least one network interface capability and may have more than one. While the network interface capability allows for a component to be connected to a communication network, it does not provide the communication (data transferring) capability. Some examples of network interface capabilities include Ethernet adapters, LTE radios [IIC-DCE] The Industrial Internet of Things Distributed Computing in the Edge, version 1.1, 2020

www.iiconsortium.org/pdf/IIoT-Distributed-Computing-in-the-Edge.pdf

[IIC-DTC] Digital Twin Core—Essential Elements for Interoperability, version 1.0, 2022

www.iiconsortium.org/foundational-publications/

[LTE-4G], ZigBee radios [CSA-ZigBee], and WiFi dongles [IEEE-WiFi].

Supporting capabilities provide additional functionality that supports the IIoT system. Examples of supporting capabilities include time synchronization, data encryption, authentication, orchestration and remote component management. Note that some IIoT components may only provide a supporting capability such as orchestration and not offer any transducer or data capabilities.

Latent capabilities are capabilities that the IIoT component could provide but are not currently enabled for access externally from the IIoT component. For example, a component may have an empty USB port with nothing plugged into it. In that state, the USB port is considered a latent capability. It has the potential to be used at any time, and if someone attaches something to it, that could enable any of the other capabilities—if someone plugs a WiFi adapter into the USB port, the IIoT component would then have an additional network interface capability. USB ports and other communication interfaces, such as serial, High-Definition Multimedia Interface (HDMI), Digital Visual Interface (DVI), DisplayPort, and External Serial Advanced Technology Attachment (eSATA), often change their state and may switch from being a latent capability to an active capability and back.

6.2.4 KEY CAPABILITY TRANSFORMATIONS

To provide benefit to an IIoT system, an IIoT component may perform some type of transformation. Key capabilities and their respective transformations are listed in Table 6.2-1:.

Capability Type	Input Type	Transform Input	Transform Output	Output Type
Sensing	Physical energy	Property of physical system state	Representation of property of physical state	Digital data
Actuating	Digital data	Representation of desired change in aspect of physical state	Changed property of physical system state	Physical energy
Data Processing	Digital data	Set of information	New set of information	Digital data
Data Storing	Digital data	Set of information	Set or subset of information available over time	Digital data
Data Transferring	Digital data	Set of information	Same set of information available over distance	Digital data

Table 6.2-1: Key Capability Transformations

6.3 THREE-TIER ARCHITECTURE PATTERN

The three-tier architecture pattern comprises edge, platform and enterprise tiers. These tiers play specific roles in processing the data flows and control flows (see chapter 5) involved in usage *activities*. They are connected by three networks, as shown in Figure 6.3-1.

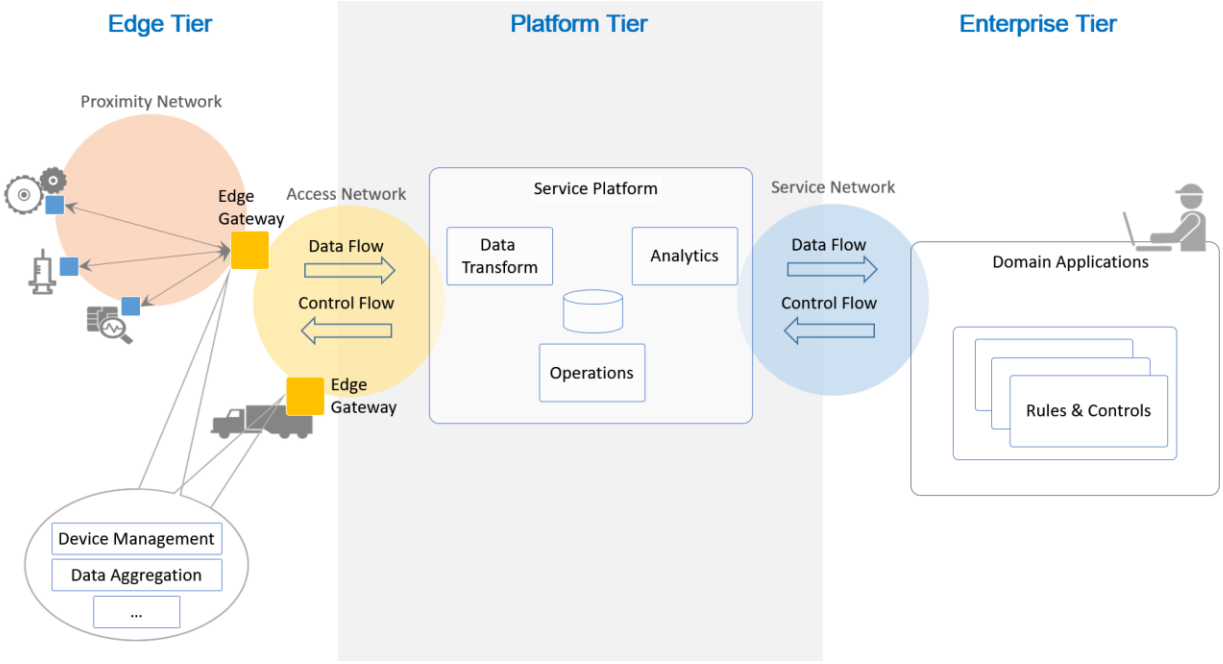


Figure 6.3-1: Three-Tier IIoT System Architecture

The *edge tier* collects data from the edge nodes, using the proximity network. The architectural characteristics of this tier, including the breadth of distribution, location, governance scope and the nature of the proximity network, vary depending on the specific use cases.

The *platform tier* receives, processes and forwards control commands from the enterprise tier to the edge tier. It consolidates processes and analyzes data flows from the edge tier and other tiers. It provides management functions for devices and assets. It also offers non-domain specific services such as data query and analytics.

The *enterprise tier* implements domain-specific applications, decision-support systems and provides interfaces to end-users including operation specialists. It receives data flows from the edge and platform tier. It also issues control commands to the platform and edge tiers.


Note

In the above figure, functional blocks are shown in each tier. These functional blocks are indicative of the primary functional location of the tier yet are not exclusively assigned to that tier. For example, the 'data transform' function in the platform tier could also be found in the edge tier (e.g. performed by a gateway) although it would be implemented in a different way and for a different purpose. For example, 'data transform' at the edge is typically done in a device-specific manner through device-specific configuration and interfaces, unlike the platform tier where it is usually supported as a higher-level service that operates on data that has been abstracted from any device source or type.

Different networks connect the tiers:

The *proximity network* connects the sensors, actuators, devices, control systems and assets, collectively called edge nodes. It typically connects these edge nodes, as one or more clusters related to a gateway that bridges to other networks.

The *access network* enables connectivity for data and control flows between the edge and the platform tiers. For example, it could be a corporate network, an overlay private network over the public internet or a 4G/5G network.

Service network enables connectivity between the services in the platform tier and the enterprise tier, and the services within each tier. It may be an overlay private network over the public Internet or the Internet itself, allowing the enterprise grade of security between end-users and various services.

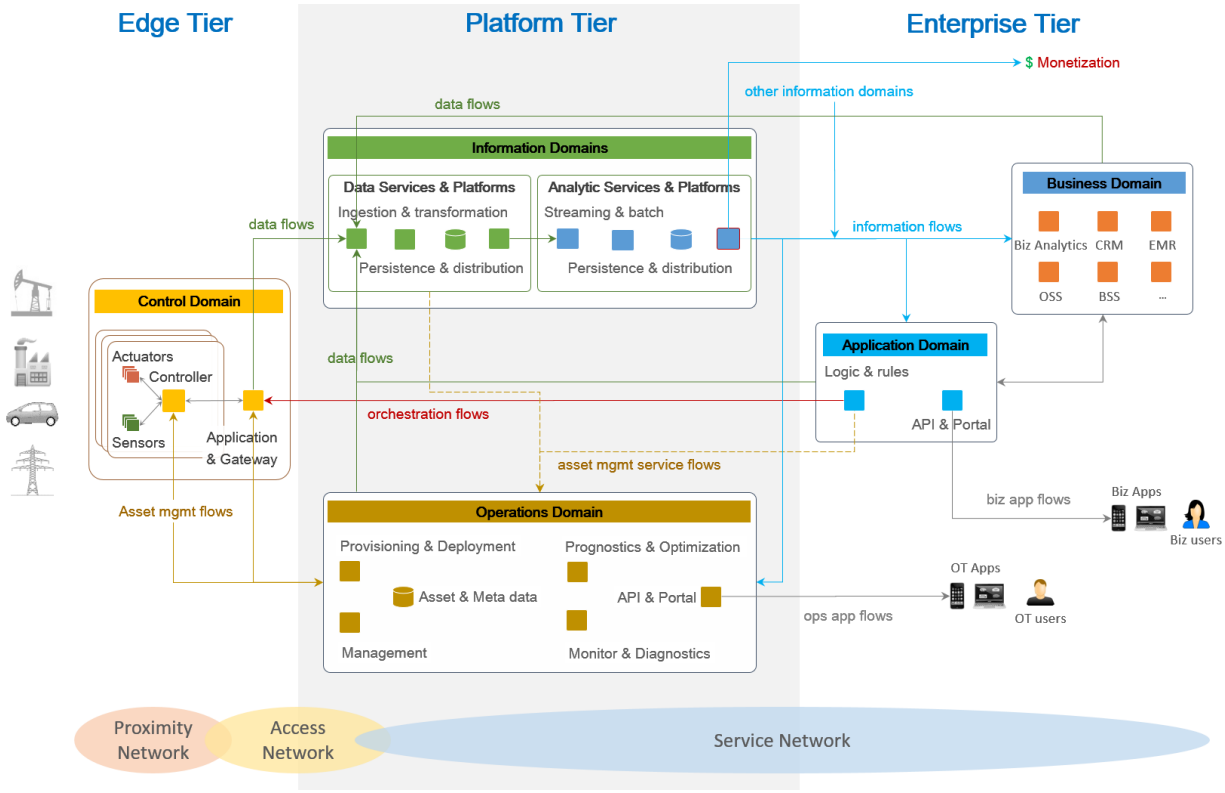


Figure 6.3-2: Mapping between a three-tier architecture to the functional domains

The three-tier architecture pattern combines major components (e.g. platforms, management services, applications) that generally map to the functional domains as shown in Figure 6.3-2. From the tier and domain perspective, the edge tier implements most of the control domain; the platform tier most of the information and operations domains; the enterprise tier most of the application and business domains. This mapping demonstrates a simple functional partitioning across tiers. In a real system, the functional mapping of IIoT system tiers depends greatly on the specifics of the system use cases and requirements. For example, some functions of the information domain may be implemented in or close to the edge tier, along with some application logic and rules to enable intelligent edge computing.

Another reason why implementation tiers do not generally have an exclusive mapping to a particular functional domain is that these tiers often provide services to each other to complete the end-to-end activities of the system. These services, for example, data analytics from the information functional domain, then become supportive of other functional domains in other tiers.

The *asset management flows* (see Figure 6.3-2) is an expression of the operations domain component of the platform tier to manage the assets in the edge tier.

The operations domain component of the platform tier itself provides services (asset management service flows) to other components, either in the same tier or in another.


Example

The data services (information domain) component of the platform tier may request services from the operations domain component for the verification of asset credentials it receives in the data flows from the edge tier, and query of asset metadata so it can augment the data received from the assets before the data are persisted or fed into analytics in the next stage of processing.

Similar operations domain services can be provided to the application domain components in the enterprise tier as well. Conversely, the operations domain components may use data services from the information domain component in order to get better intelligence from asset data, e.g. for diagnostics, prognostics and optimization on the assets.

As a result, components from all functional domains may leverage the same data and use analytic platforms and services to transform data into information for their specific purposes.

6.4 Gateway-Mediated Edge Connectivity and Management architecture pattern

The gateway-mediated edge connectivity and management architecture pattern comprises a local connectivity solution for the edge of an IIoT system, with a gateway that bridges to a wide area network as shown in Figure 6.4-1. The gateway acts as an endpoint for the wide area network while isolating the local network of edge nodes. This architecture pattern allows for localizing operations and controls (edge analytics and computing). Its main benefit is in breaking down the complexity of IIoT systems, so that they may scale up both in numbers of managed assets and in networking. However, it may not be suited to systems where assets are mobile in a way that does not allow for stable clusters within the local network boundaries.

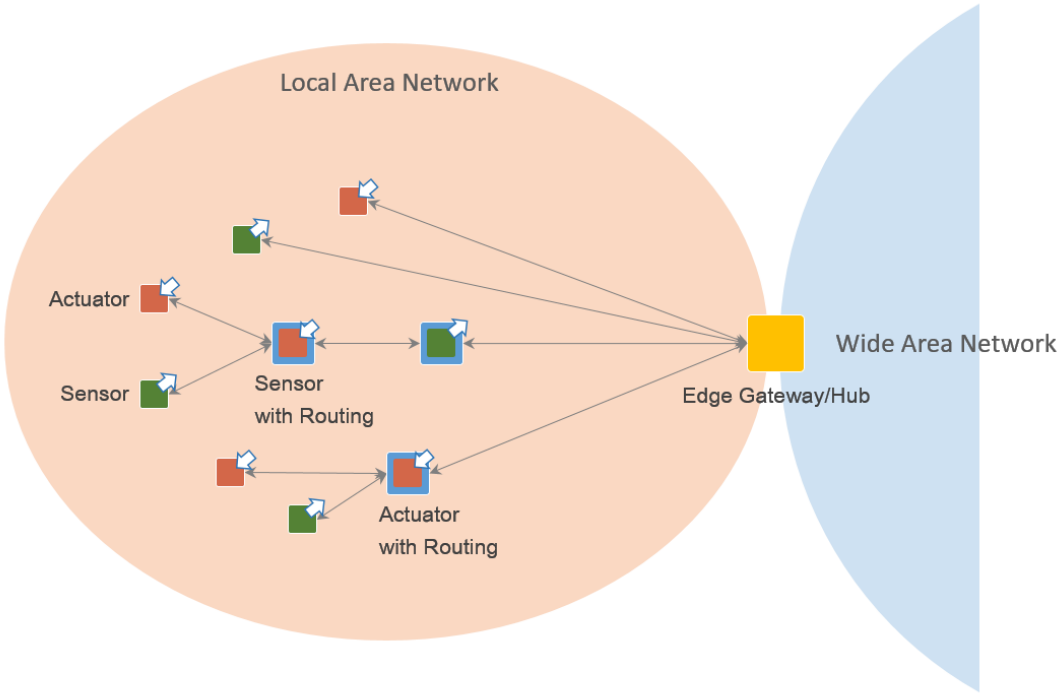


Figure 6.4-1: Gateway-Mediated Edge Connectivity and Management Pattern

The edge gateway may also be used as a management point for devices and assets and data aggregation point where some data processing, analytics and control logic are locally deployed.

The local network may use different topologies as described below:

In a *hub-and-spoke* topology, an edge gateway acts as a hub for connecting a cluster of edge nodes to each other and to a wide area network. It has a direct connection to each edge entity in the cluster allowing in-flow data from the edge nodes, and out-flow control commands to the edge nodes.

In a *mesh network* (or peer-to-peer) topology, an edge gateway also acts as a hub for connecting a cluster of edge nodes to a wide area network. In this topology, however, some of the edge nodes have routing capability. As result, the routing paths from an edge node to another and to the edge gateway vary and may change dynamically. This topology is best suited to provide broad area coverage for low-power and low-data rate applications on resource-constrained devices that are geographically distributed.

In both topologies, the edge nodes are not directly accessible from the wide area network. The edge gateway acts as the single-entry point to the edge nodes and as management point providing routing and address translation.

The edge gateway supports the following capabilities:

- *Local connectivity* through wired serial buses and short-range wireless networks. New communication technologies and protocols are emerging in new deployments.
- *Network and protocol bridging* supporting various data transfer modes between the edge nodes and the wide area network: asynchronous, streaming, event-based and store-and-forward.
- *Local data processing* including aggregation, transformation, filtering, consolidation and analytics.
- *Device and asset control and management point* that manages the edge nodes locally and acts an agent enabling remote management of the edge nodes via the wide area network and
- *Site-specific decision and application logic* that are performed within the local scope.

6.5 DIGITAL TWIN CORE AS A MIDDLEWARE ARCHITECTURE PATTERN

The industrial internet primarily deals with entities in the real world, most significantly industrial equipment in production and operational environments. Through the internet of things, the industrial internet seeks to connect to industrial equipment, collect data from them, and apply advanced analytics to the data to gain insights to optimize production and operation processes. It has become increasingly apparent that systematic approaches are needed to organize and manage the data to support effective data analytics for industrial environments where there may be hundreds of pieces of equipment, each with possibly hundreds of data points and each piece of equipment having complex relations to other equipment and products under production. Digital twins have come about as a technology that provides a digital representation of real-world entities, including real-time and historical states and behaviors. It is on these digital models of the corresponding real-world entities that advanced analytics can be effectively applied to provide insights to drive optimization of production and operation processes.

To facilitate a new generation of industrial applications that are enabled by the digital twins, new technologies, frameworks, or even new architectures may be needed. To start, many of such application systems in the industrial operational environments (industrial applications for short) require data synchronization between the digital twins and their real-world counterparts. A system with digital twins connects to the equipment and collects data from them, preprocesses, stores and manages these data. It also requires the capability for establishing the data models, running computation models, and providing services to access these data and computation results in digital twins. Additionally, it requires all other aspects that a conventional application would require ranging from installation, configuration, security, management to DevOps.

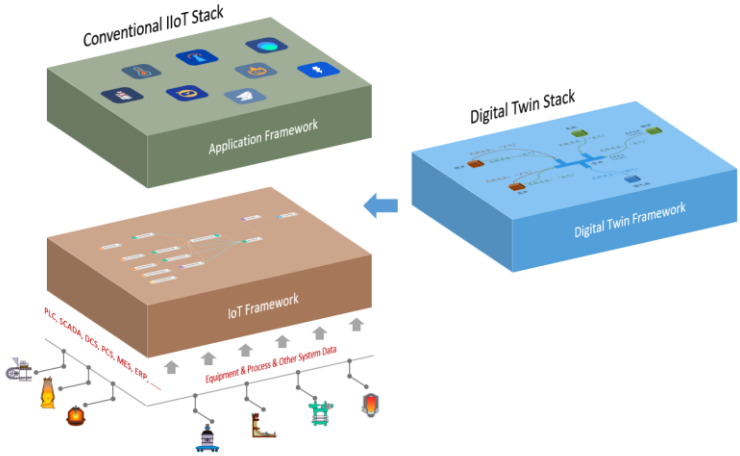


Figure 6.5-1: Combining digital twin with IIoT

With the prevailing application DevOps trend, it is increasingly common for industrial applications to be implemented based on certain technology platforms, including some industrial platform-as-a-service (PaaS). These platforms provide connectivity to equipment and real-world environments (through sensors), perform data collection, pre-processing, storage, and management, and enable application DevOps with micro-service support. The purpose of these platforms is to enable and simplify application development by providing proven system architecture and ready-to-use common functionalities as platform services thus reducing the complexity and effort in application development. Take for example, a common IIoT application platform may combine an IIoT (connectivity, data management, etc. functionalities) framework with an application DevOps framework. As digital twin support becomes widely available, it is natural to provide a digital twin framework as part of the platform service, that is supported by the IIoT services and in turn supports the industrial applications. In a full-stack architecture for IIoT applications that leverage digital twin capabilities, the core capabilities of digital twins can be implemented as a middle layer, supporting individual applications in the upper applications layer, and relying on various supportive technologies and platforms provided in the lower support environment layer. Examples of support functions include equipment connectivity, data collection, preprocessing and storage that can be readily provided by an IIoT system.

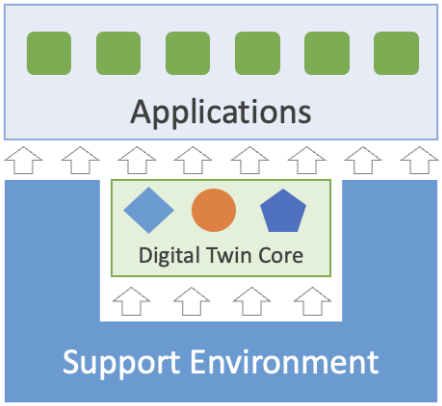


Figure 6.5-2: Digital Twin Core as a middleware layer

With a digital twin core middleware architecture, standardization of the complete architecture stack to support interoperability of digital twins is not necessary. It simplifies enabling interoperability of digital twins. A standard digital twin core, with its reduced scope and complexity making it more versatile, can be more easily supported by various industrial application systems, regardless their architecture details and technology choices. Consequently, interoperability can be more widely supported. Moreover, the notion of a digital twin core middleware also makes it more flexible to build industrial applications with digital twin technologies. This is because there are many options and variations in architecture and technologies over which the industrial applications can be built. Moreover, the architectures and technologies are evolving rapidly. The separation through loosely coupled design between the digital twin core capability with the architecture and technologies in which it is embedded makes it easier to adapt digital twin capabilities in a standard way to many architectures and to have a large choices of technologies to implement these capabilities.

With a digital twin core as middleware, we can conceptualize a general three-layer application architecture with an upper, middle and lower layers, application, digital twin core and support environment, respectively.

- The application layer is where the business logics of the application is implemented. In solving specific operational problems, the application layer would require information and analytics about the states and behaviors of the real-world entities that are to be provided by the middle layer below.
- The digital twin core implements the digital representation of the real-world. For detailed information of the important functions and their structure of digital twin core, refer to the IIC Technical Report, “Digital Twin Core: Essential Elements for Interoperability” [IIC-DTC].

- The support environment layer includes technological frameworks that support the middle- and upper-layer components. From the supporting digital twin core perspective, it may include a number of other supportive technologies described below.

The supportive environment provide:

IoT framework: To realize the representation of the real-world live, the digital twin core requires communication with the real-world entities to gather data and send commands to effect changes. Bi-directional connectivity to the real-world entities is commonly implemented as an IoT system or subsystems.

Scale-out database framework: The digital twin core requires various databases to store and query data about the real-world. These data are of diverse types, from structural to unstructured data and from timeseries to streaming media. Some data can be voluminous, for example, timeseries data about the state of the real-world entities, covering historical, current (real time) and predicted data continuum, from equipment with fast-moving components that require high-frequency data collection and computation. Therefore, some sort of distributed scale-out database with horizontal scalability may be in good order.

Modern computing runtime framework: The digital twin core would benefit from modern computing environments for it to be deployed, run and managed. The same framework can also support components in the application layer. In this regard, modern virtualized and distributed, better yet cloud-native computing runtime environment such as containerized application management frameworks, e.g. Docker managed Kubernetes. These virtualized, cloud-native and distributed computing frameworks allow for horizontal scaling of computing resources to meet the increasing need of computation for data synchronization between the applications with the real-world in both volume and fidelity, and more computation-demanding advanced analytics. The containerized environment would make it easier to deploy, run and manage the core services of the digital twin core, along with other services and functions for the overall industrial applications.

Computation acceleration framework: The computation models in the digital twin core associated with specific equipment are the key elements that generate key values for entire applications. The execution of these computation models will increasingly demand on large computing capabilities, for example Graphic Processing Unit (GPU) and Neural Computing Unit (NPU) in additional to the common Central Processing Unit (CPU), to run and accelerate advanced analytics involving machine learning and AI algorithms.

Machine-learning and AI framework: the computation models in the digital twin core will increasingly involve machine-learning and AI algorithms. A framework supporting common machine-learning and AI algorithms with toolsets supporting the full-lifecycle of model building and execution including data exploration, cleansing and preprocessing, model exploration,

building, validating, running will become more important to the success of digital twin applications.

Physical- and system-simulation framework: Simulation models are a frequently used modeling technique in the digital twin core to perform what-if analysis, diagnostics, predictions and simulation-based optimizations. A framework that supports physical and system modeling and simulation would ensure the effectiveness and trustworthiness of the modeling, synchronization and integration of digital twins. Toolsets and techniques in the framework support the whole simulation lifecycle, i.e. requirements, development, deployment, use and maintenance and retirement. Verification, validation, and uncertainty quantification (VVUQ) standards and techniques may be selected to ensure the simulation to stay credible and trustworthy throughout its lifecycle. It should not only be about the simulation itself, but also include the interactions between simulations with the physical counterpart, with other relevant digital twin components (e.g. data processing, visualization, GUI and control), and other existing enterprise applications (MES, ERP, PLM).

Modern Dev-Ops framework: Industrial applications that rely on data analytics require continuing improvements. On the other hand, with the increasing demand for flexible manufacturing capabilities to support more varieties of smaller batch production, the corresponding industrial applications that manage and optimize the manufacturing processes will need to be more flexible to adapt and respond to the dynamic change of requirements on the manufacturing floor. Therefore, a flexible development and operation process to accommodate enhancements of data analytics and changes of functional requirements in the digital industrial applications is needed. Modern Dev-Ops processes, methodologies, tools and frameworks streamline the collaboration between business, development and operations teams to deliver high-quality software solutions swiftly. Combined with agile development processes, it can simplify the processes from requirement, to design, development and quality assurance and then automate the processes from requirements to build to deployment of a software application. In some cases, it enables controlled and limited tests in the production environment and a graduate rolling-out of updates to minimizing the impact to the production environment. The ability to adopt these set of new approaches and technologies to the industrial environment and how to bridge the gap between this new approach versus the traditional waterfall application development and operation processes that have been customarily practiced in the industries will be important.

Three-dimensional visualization, virtual reality and augmented reality frameworks: Three-dimensional (3D) visualization deals with how to render the real-world objects in computer imitating the spatial (shape and form), material (surface texture and lighting), movement (physical law) characteristics of these objects. The three-dimension visualization technologies have advanced a great deal the recent decades, in some way propelled by the computer gaming industries, evolving from static single object 3D simple rendering such a part of a machine in a CAD program to more complex, dynamic, and ever realistic objects rendering in contextual environments, giving rise to virtual reality (VR), augmented reality (AR) and now metaverse.

These technologies are relevant and valuable to aid intuitive human comprehension of the real-world, including machines and parts. Since they are a representation of the real world, they are inevitably an important capability of digital twins. These technologies are not digital twin themselves but tools the digital twin can leverage to enhance human comprehension of the real-world. It can be envisioned that the support environment provides various respective 3D rendering, VR and AR frameworks and the digital twin core contains various model definitions for each of the real-world entities it represents. These models can then be readily rendered by their respective frameworks on demand in the applications.

The capabilities from various technological frameworks described above can be offered to or be used, as services through some service wrappers, by the components in the digital twin core and the application layers. For details about the digital twin core and its metamodels, refer to the IIC Digital Twin Core Technical Report (Reference to be added).

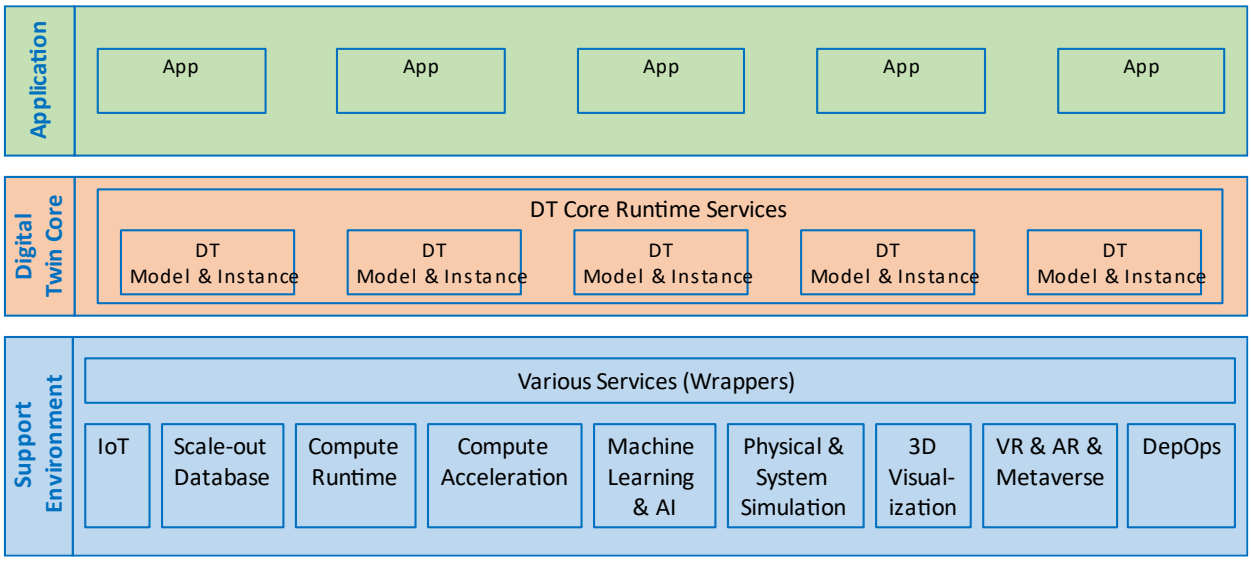


Figure 6.5-3: An Illustration of Key Technologies Supporting Digital Twin Core and Industrial Applications in a Three-Tier Architecture with Digital Twin Core as a Middleware

6.6 LAYERED DATABUS ARCHITECTURE PATTERN

The layered databus is a common architecture across IIoT systems in multiple industries (see Figure 6.6-1 below). This architecture provides low-latency, secure, peer-to-peer data communications across logical layers of the system. It is most useful for systems that must manage direct interactions between applications in the field, such as control, local monitoring and edge analytics.

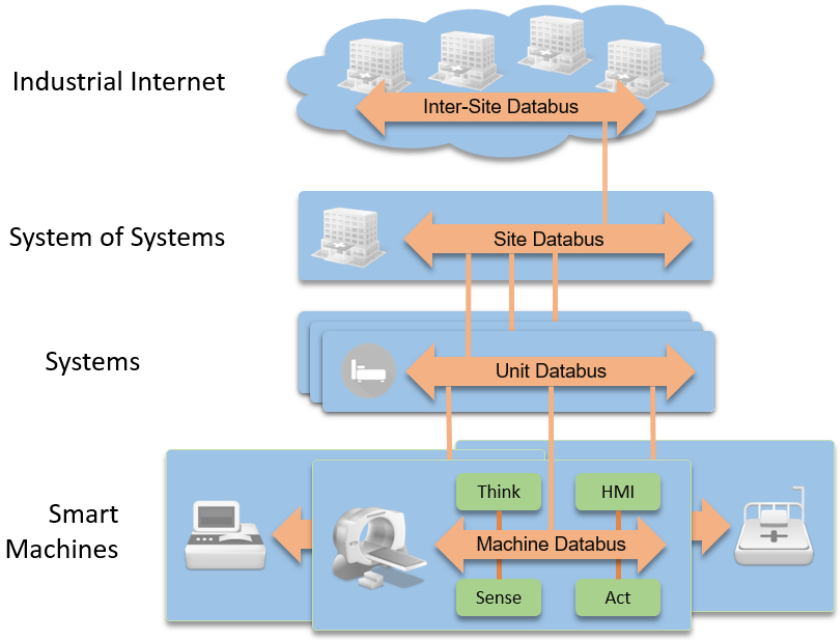


Figure 6.6-1: Layered Databus Architecture

At the lowest level, smart machines use databuses for local control, automation, and real-time analytics. Higher-level systems use another databus for supervisory control and monitoring. Federating these systems into a "system of systems" enables complex, internet-scale, potentially-cloud-based, control, monitoring and analytic applications.

A databus is a logical connector that implements a set of common schemas and communicates using those set of schemas between endpoints. Each layer of the databus implements a common data model, allowing interoperable communications between endpoints at that layer.

The databus supports communication between applications and devices. For instance, a databus can be deployed within a smart machine to connect its internal sensors, actuators, controls and analytics. At a higher level, another databus can be used for communications between machines. At a system-of-systems level, a different databus can connect a series of systems for coordinated control, monitoring and analysis. Each databus may have a different schema or data model. Data models change between layers, as lower-level databuses export a controlled set of internal data.

Adapters may be used between layers to match data models. The adapters may also separate and bridge security domains, or act as interface points for integrating legacy systems or different protocols.

Generally, transitions that occur between layers filter and reduce the data. The scope of control and analysis increases at each layer and the amount of data is reduced to match the broader scope, higher latencies and higher level of abstraction. An example use of this architecture for oil well monitoring and operational control, typical for large SCADA systems [NCS-SCADA], is shown in Figure 6.6-2.

In addition to its use in the control, information, application and enterprise domains, this layered databus architecture is also useful in the operations domain for monitoring, provisioning and managing devices, applications and subsystems within the system.

Central to the databus is a data-centric publish-subscribe communications model. Applications on the databus simply “subscribe” to data they need and “publish” information they produce. Messages logically pass directly between the communicating nodes. The fundamental communications model implies both discovery—what data should be sent—and delivery—when and where to send it. This design mirrors time-critical information delivery systems in everyday life including television, radio, magazines, and newspapers. Publish-subscribe systems are effective at distributing large quantities of time-critical information quickly, especially in the presence of unreliable delivery mechanisms.

The layered databus architecture offers the following benefits:

- fast device-to-device integration, with delivery times in milliseconds or microseconds,
- automatic data and application discovery within and between busses,
- scalable integration, comprising hundreds of thousands of sensors and actuators,
- natural redundancy, allowing extreme availability and
- hierarchical subsystem isolation, enabling development of complex system designs.

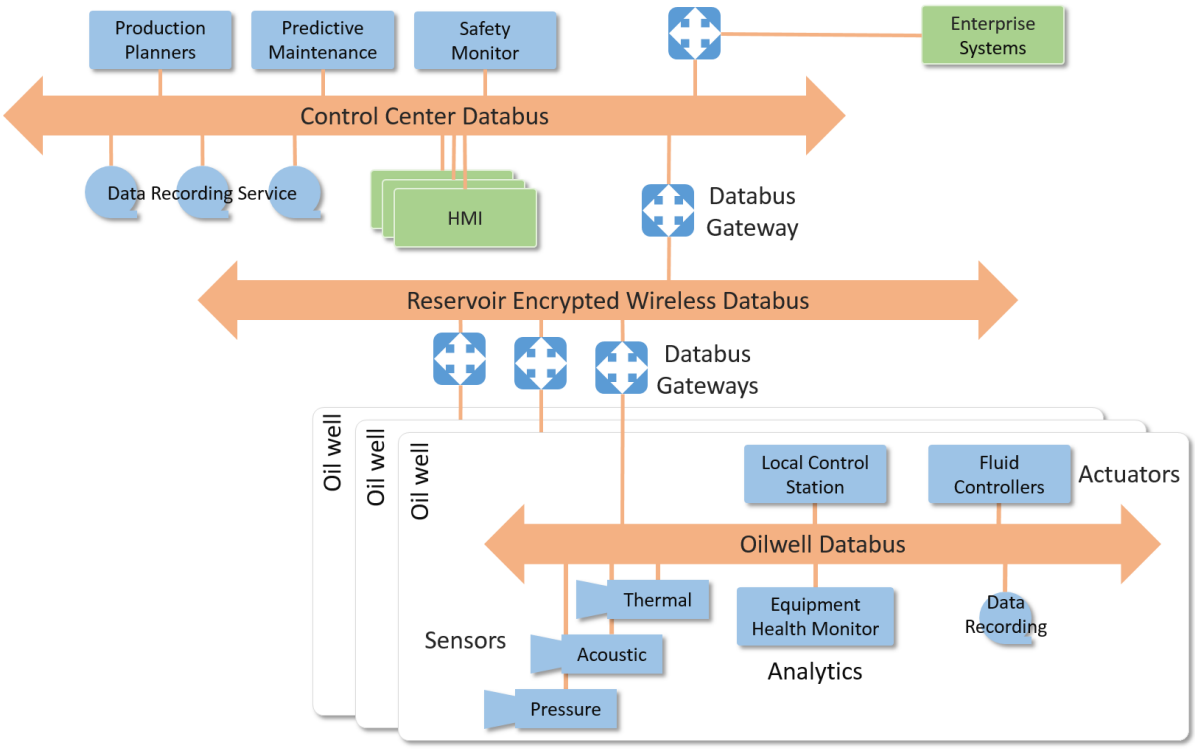


Figure 6.6-2: A three-layer databus architecture.[1]

6.7 System-of-Systems Orchestrator Architecture Pattern

As IIoT systems have matured, they have created opportunities to pool their resources and capabilities together to create new, more complex systems that offer more performance, functionality and overall benefits than the sum of systems by themselves. The system-of-systems (SoS) orchestrator pattern is a collection of systems, each capable of independent operation, that interoperate together to achieve additional desired capabilities.

At a high level, the IIoT system may be configured as one or more distributed, independent constituent systems, connected to a single SoS Orchestrator, which is connected to one or more SoS Services, creating a system-of-systems (see Figure 6.7-1:).

[1] Note that the control center HMIs can access any sensor value from the Oil Well databuses.

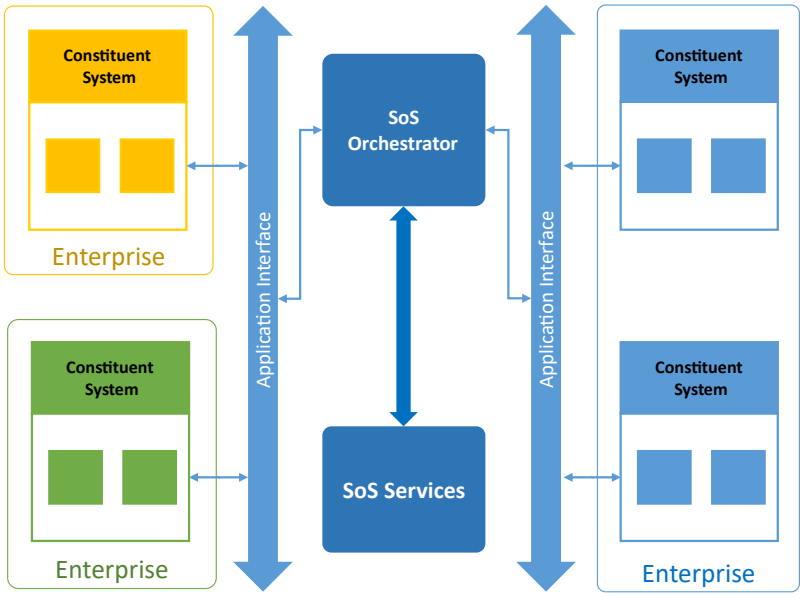


Figure 6.7-1: A System of Systems Architecture

The central characteristic of a system-of-systems is that they form from a variety of constituent systems: existing or legacy systems, newly engineered from the "ground-up" custom systems and potentially tailored existing commercial-off-the-shelf (COTS) systems. In addition, five additional characteristics [ISO-21839] can be found in most system-of-systems designs:

1. Operational independence of the individual systems
    - The constituent systems within the system-of-systems achieve well substantiated purposes by themselves and continue to operate in this way and accomplish their individual purposes even if detached from the overall system.
2. Managerial independence of the systems
    - The constituent systems are managed in large part for their own purposes rather than the purposes of the whole.
3. Geographic distribution
    - The constituent systems are also most likely located or dispersed in various geographies—within a factory, a city, state, country or even worldwide.
4. Emergent behavior
    - While meeting the above constituent system needs, the system-of-systems enables new capabilities that are generally unachievable by the individual system acting independently.
5. Evolutionary development
    - A system-of-system may not appear fully formed and functional at the start. Its development is evolutionary in the sense that functions and purposes are added, removed and iterated, with experience in use of the system-of-systems.

One key benefit of a system-of-systems is its emergent capabilities, in what are called SoS services. These are services, offered to end users and other external systems, that constituent

systems cannot offer on their own. To achieve these emergent capabilities, the collective operation of constituent systems must be coordinated.

Since constituent systems operate with no inherent dependencies to each other, their coordination is performed by the SoS orchestrator. The SoS orchestrator communicates and coordinates with the constituent systems over a service bus. The SoS orchestrator also gathers data from the constituent systems and provides the collected data to one or more SoS services, which provide end user interfaces. SoS services take the collective resources of the constituent systems and offer new capabilities to end users and other external systems. SoS services offer emergent capabilities to end users and other external systems; these are capabilities that the constituent systems cannot offer independently.

Constituent systems can belong to the same or different enterprises. They are managed independently, even when they belong to the same enterprise. Since constituent systems operate with no inherent dependencies on each other and may not be aware of each other's existence, their coordination is performed by the SoS orchestrator, which then connects to one or more SoS services.

The *SoS orchestrator* communicates and coordinates with the constituent systems over an application interface. The SoS orchestrator also gathers data from the constituent systems and provides insight acquired by analyzing this data to one or more SoS services. Because of the managerial and operational independence of each constituent system, it is the SoS orchestrator's role to communicate between them. The relationships between an SoS orchestrator to any constituent system is loosely coupled, since it is most likely that constituent systems are unaware of the existence of a SoS orchestrator. Therefore, communication is generally a publisher-subscriber model.

SoS services take the collective resources of the constituent systems and offer new emergent behaviors in the form of additional capabilities. They provide end-user interfaces, allowing these capabilities to be consumed by end users and other external systems (i.e. outside the SoS). The relationship between the SoS orchestrator and SoS service is tightly coupled, therefore client-server communication models may be more appropriate.

At the logical level, the pattern allows only one SoS orchestrator in each system-of-systems. All coordination between constituent systems and SoS services is handled by the SoS orchestrator. Data moves between a constituent System and the SoS orchestrator, and between an SoS service and the SoS orchestrator. The SoS orchestrator is a gatekeeper, allowing constituent systems to maintain their independence from each other while restricting them from having direct access to SoS services.

There are two main subtypes: vertical system-of-systems and horizontal system-of-systems. The major difference between the two subtypes is whether the constituent systems belong to a single enterprise or to multiple enterprises.

A vertical system-of-systems contains independent constituent systems, each of which belong to the same enterprise. Although each constituent system has managerial independence from each other, they will share common enterprise-level concerns and tools, for example an enterprise-wide database. One area of common concern is security; another is enterprise objectives. The SoS orchestrator supervises each constituent system and manages a common set of shared enterprise objectives. A typical use case for a vertical SoS would be offering workload balancing or resource optimization services for disparate systems.

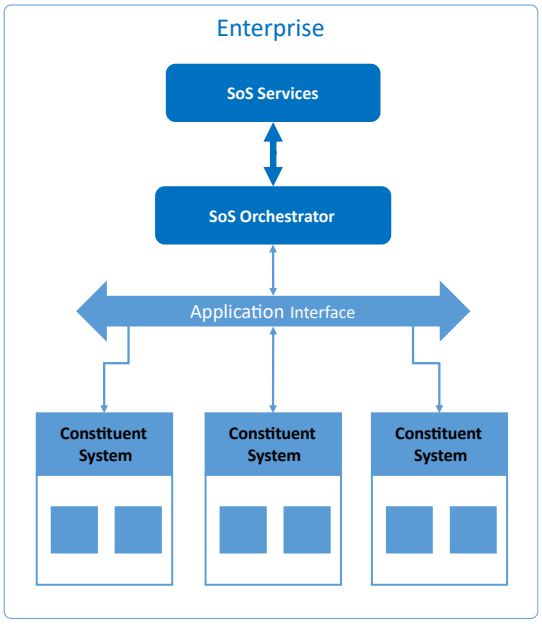


Figure 6.7-2: A Vertical System of Systems

A horizontal system-of-systems contains independent constituent systems that belong to different enterprises or industries. The SoS orchestrator coordinates the communication of each constituent system. The main activity of the SoS orchestrator is mediation for each system whose enterprise objectives may be different. For example, the SoS orchestrator and SoS services may be in an internet environment and not part of any constituent system enterprise. In this case, each system has its own independent security realm. A typical use case for a horizontal SoS would be correlating information gathered from each constituent system to provide new business services to other enterprises.

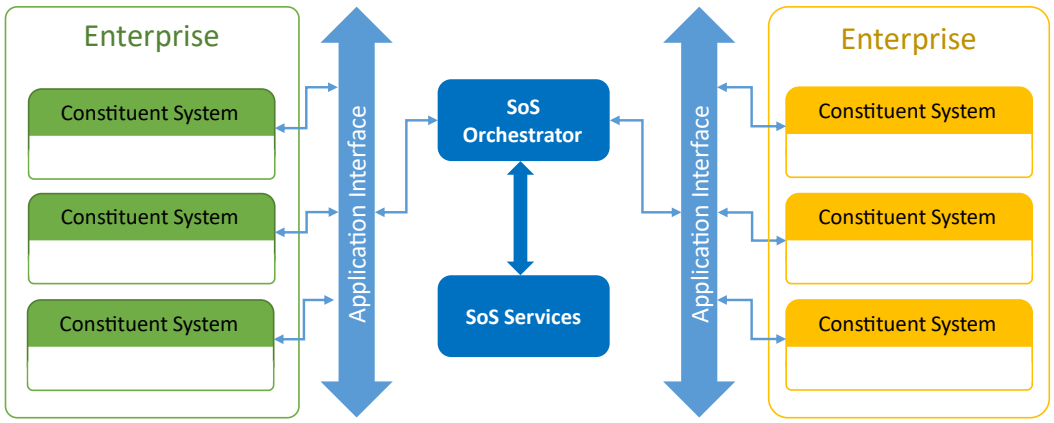


Figure 6.7-3: A Horizontal System of Systems

Combining constituent systems to form a system-of-systems impacts each viewpoint in the IIRA, and new sets of concerns may be raised by the system-of-systems that were not original to the constituent systems by themselves. The new concerns can have a cascading and comprehensive impact on both the system-of-systems itself and the individual constituent systems, especially in terms of system characteristics and trustworthiness. System-of-systems, while not necessarily large, are by nature complex, providing higher levels of sophistication and efficiency. The resulting system should be evaluated by its own set of metrics, and not just the net combination of the constituent system's metrics [Maier-1999].

Example

Virtual Power Plant (VPP)

The Virtual Power Plant manages and orchestrates the power generation of various electrical power and distribution systems, such as coal-fired, photovoltaic and wind. By combining the power generation of each system together, the electric grid can become more adaptive to demand response needs and more stable as renewable sources power output fluctuates.

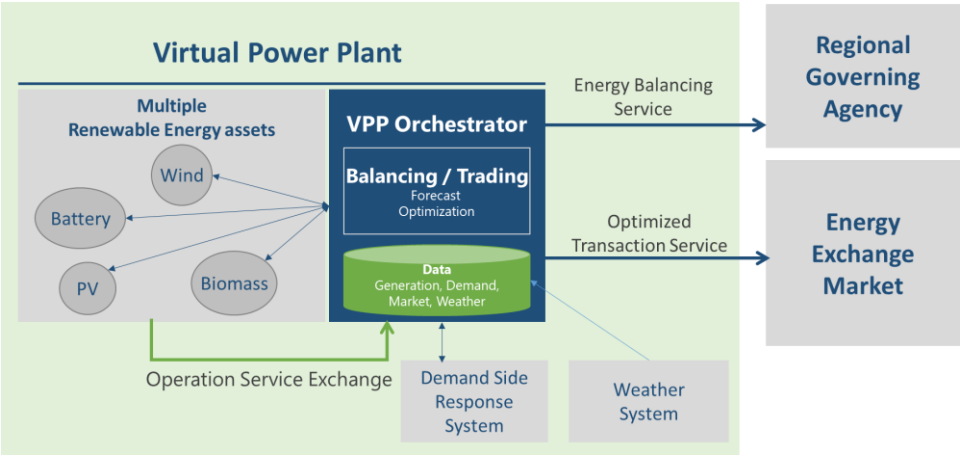


Figure 6.7-4: Virtual Power Plant System of Systems

Annex A DESIGN SPACE CONSIDERATIONS

Architecture and high-level design are about making consistent choices in a design space, the full range design possibilities spanned by the domains of its multitudes of design parameters, and the multiple combinations of choices in that space remains for the most part unexplored. Indeed, a consequence of the size of the design space for IIoT systems is that it will remain for the most part unexplored. When applying the Industrial Internet Reference Architecture to real-world IIoT systems, it would be beneficial to have a broad view of possible design parameters and their constraints in identifying, describing, and resolving system concerns. For this purpose, we include these design space considerations as an Appendix, and we may further develop and refine these considerations in further revisions to making it more useful for the system architects. The specific design exemplars illustrated here are not intended to be proscriptive, and we encourage exploration of the design space for new and consequential combinations that will lead to surprising capabilities and applications. The table is intended to be illustrative.

Topics	Variations			
Location awareness	Applications cannot tell where they are running, unless they can infer from timestamps on data (e.g. that after reading a sensor, the data is several seconds old on arrival)	Applications can tell what clique they are in, and type of device, e.g. e.g. control tier, embedded in turbine; enterprise tier, virtual on set of servers owned by client	Applications can be tied to specific types of devices, have access to maps and can infer physical location properties of devices	Applications know exactly in the 4-dimension space (time-space) where they are executing, what hardware they are near and what paths are available by specific route to any local device
Communication paradigm	Stigmergy only (behavior must be observed through environmental changes)	Ad hoc port-based protocols (e.g. point-to-point API based communication)	Pub-sub (having some known structured information)	Message passing using speech acts (with formal semantic logical forms)
Computational assignment	All devices are homogeneous	Devices have specific capabilities, but no particular constraint on where code can run	Code can only be run in appropriate clique of devices	Code can only be run on specific unique device
Execution paradigm	Ad hoc, every device/code combination unique	Data flow—data moves based on interest processing resident	Processing centric— data has specific execution/view history attached and very controlled point to point flow; processing resident	Data resident — processing moves to the data which controls access; limited information may be moved with the process (local state) [e.g. mobile agents]
Generic resource management	Systems engineer performs offline	Automated at device level only (chronologic, preemptive, …, real time)	Automated at clique level	Automated at IIoT system level
Certifications (safety, security, …)	Systems engineer performs offline	Specific sets of flows are certified offline for situations detected online	Automated ‘proof’ for a flow performed online, but ahead of engagement	Automated ‘proof’ performed during execution (e.g. deontic logic)
Addressability	Endpoint address, device name and path needed	Service name	Content addressable (e.g. all services that can perform an IR (information retrieval) observation at point (x, y, z) at time T; all services that know if P)	

Topics	Variations			
Constraint expressivity	No constraints supported	Can specify data handling (e.g. send via different path than last time using 16 new TOR servers)	Can specify processing (e.g. spend no more than 2G nominal CPU cycles on inferences on usage based on the address properties of data or the utilization of device processing for the following conversation marked 'A')	Can specify arbitrary higher order properties (e.g. "there exists a service S, that can predict the likelihood of commodity price shortages" [without specifying where S is, or if it is reachable]
Negotiability	None—take it or leave it	Over specific predetermined system-wide resources, e.g. $ to invoke a service at a fixed quality level	Enter into auction (e.g. Dutch) for specific service availability (e.g. service S has timeslot T available: bid?)	Arbitrary resource/quality negotiation and contract remediation (negotiate 2x quality for 4x time and 8x price, but only 1.5x quality produced so penalty is…)
Resilience	None—single point of failure throughout	Some reliability measures taken (failover redundancy, voting) for predetermined set of critical resources, but general system still has single points of failure (e.g. corrupted security operator)	Resilience at clique level (pool of similar resources allow loss of some without noticeable degradation, pool can be replenished from a system-wide reserve, multiple cross-checks implemented, e.g. vs. insider threat)	Individual device resilience and reconstruction (robot repairs the device and improves it so the same failure will not recur, recovers state of device at time of failure, etc.)

Table 6.7-1: Architectural Alternative/Design Space

Annex B TERMS AND DEFINITIONS

The following unique terms and definitions are used in this document:[1]

- *Concern*: any topic of interest pertaining to a system.
- *Architecture viewpoint* (viewpoint for short): conventions framing the description and analysis of specific system concerns.
- *Stakeholder*: an individual, team or organization having an interest in a concern and, by extension an interest in, the viewpoint and system.

[1] Many of these terms will be elaborated in the context where they are introduced and used.

- *Model kind*: a set of conventions describing, analyzing and resolving concerns in a specific way. A viewpoint may specify one or more model kinds.
- *Model*: the outcome of applying the conventions of a specific model kind in a viewpoint to describe, analyze and resolve a specific set of concerns in that viewpoint.
- *Architecture view*: the collection of ideas describing, analyzing and resolving the set of specific concerns in a viewpoint using the conventions set forth in that viewpoint. A view includes one or more models.
- *Reference architecture:* an architecture template for a particular application domain providing a common and consistent vocabulary, concepts, structures and approaches, with which to discuss and create concrete architectures for specific application scenarios.

Terms that require definition are rendered in *italics*. (As the usage immediately preceding demonstrates, italics may also be used as example, or for emphasis.)

Generally, only the first use of the term is italicized. However, when a term can be read in its usual English language mode, the first use of the term may be italicized as the discussion becomes technical. In the first example below, "safety" and "security" are used informally. In the second, it introduces a definition.


Example

"Among the system characteristics that must be considered, safety is perhaps the most important, followed by security."

"*Safety* is the condition of the system operating without causing unacceptable risk of physical injury or damage to the health of people, either directly or indirectly, as a result of damage to property or to the environment."

Annex C REFERENCES

[IEEE-42010] ISO/IEC/IEEE 42010:2011 Systems and software engineering -- Architecture description
www.iso.org/standard/50508.html

[IIC-IISF] Industrial Internet Security Framework Technical Document, version 1.0, 2016
www.iiconsortium.org/IISF.htm

[IITF] Industrial Internet Trustworthiness Framework Foundations, version 1.0, 2021

www.iiconsortium.org/pdf/Trustworthiness_Framework_Foundations.pdf

[IIC-IIV] Industry Internet of Things Vocabulary, version 3.0, 2022
www.iiconsortium.org/vocab

[NCS-SCADA] SP 800-82 Rev. 2, Guide to Industrial Control Systems (ICS) Security, 2015
csrc.nist.gov/publications/detail/sp/800-82/rev-2/final

[Petruz-2016] F. Petruzella: "Programmable Logic Controllers", 5th edition, McGraw-Hill Education, 2017
www.mheducation.com/highered/product/programmable-logic-controllers-petruzella/M9780073373843.html

[Lydon-2011] B. Lydon, "PLC vs. DCS - Competing Process Control Philosophy" automation.com, 2011
www.automation.com/automation-news/article/plc-vs-dcs-competing-process-control-philosophy

[Harnad-1990] Stevan Harnad: "The Symbol Grounding Problem", Princeton University, Department of Psychology 1990
arxiv.org/html/cs/9906002

[Searle-1980] John R. Searle: "Minds, Brains and Programs", Behavioral and Brain Sciences, Volume 3, Issue 3, 1980
doi.org/10.1017/S0140525X00005756

[Thomson-1985] Judith J. Thomson: "The Trolley Problem", The Yale Law Journal, Vol. 94, No. 6 (May, 1985), pp. 1395-1415, The Yale Law Journal Company, Inc.
www.jstor.org/stable/i232661

[OMG-BMM] Object Management Group: "Business Motivation Model (BMM)", 2015
www.omg.org/spec/BMM/

[OMG-SACM] Object Management Group: "Structured Assurance Case Metamodel (SACM)", 2021
www.omg.org/spec/SACM

[IIC-KSC] Industrial Internet Consortium: "Key System Concerns, version 1.0", 2018
www.iiconsortium.org/pdf/Industrial_Internet_of_Things_Volume_G2-Key_System_Concerns_2018_08_07.pdf

[ISO-21839] ISO/IEC/IEEE 21839:2019 Systems and software engineering—System of systems (SoS) considerations in life cycle stages of a system
www.iso.org/standard/71955.html

[Maier-1999] Mark W. Maier, Architecting Principles for Systems-of-Systems, 1999
doi.org/10.1002/(SICI)1520-6858(1998)1:4%3C267::AID-SYS3%3E3.0.CO;2-D

[OPC-UA] IEC TR 62541-1:2020 OPC Unified Architecture - Part 1: Overview and concepts, 2020
webstore.iec.ch/publication/61109

[OPC-DA] OPC Data Access Custom Interface Standard, v2.05, 2012
opcfoundation.org/developer-tools/specifications-classic/data-access/

[MODBUS] MODBUS Application Protocol Specification, v1.1b3, 2012
modbus.org/specs.php

[MQTT] ISO/IEC 20922:2016 Information technology—Message Queuing Telemetry Transport (MQTT) v3.1.1
www.iso.org/standard/69466.html

[IIC-vPLC] Virtualized Programmable Logic Controllers, IIC Tech Brief, 2022
www.iiconsortium.org/wp-content/uploads/sites/2/2022/05/IIC-Edge-vPLC-Tech-Brief-20210907.pdf

[IIC-DCE] The Industrial Internet of Things Distributed Computing in the Edge, version 1.1, 2020

www.iiconsortium.org/pdf/IIoT-Distributed-Computing-in-the-Edge.pdf

[IIC-DTC] Digital Twin Core—Essential Elements for Interoperability, version 1.0, 2022

www.iiconsortium.org/foundational-publications/

[LTE-4G] Long-Term Evolution wireless broadband communications, 3GPP

www.3gpp.org/specifications/specifications

[IEEE-WiFi] IEEE 802 Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2020

standards.ieee.org/ieee/802.11/7028/

[CSA-ZigBee] Zigbee Specifications, Connectivity Standards Alliance

csa-iot.org/all-solutions/zigbee/

Annex D REVISION HISTORY

Revision	Date	Editor	Changes Made
V1.7	2015-06-04	Shi-Wan Lin (Lead, Intel), Stephen Mellor (Lead, IIC), Bradford Miller (Lead, V1.5, GE), Jacques Durand (Fujitsu), Mark Crawford (SAP), and Robert Lembree (V1.5, Intel).	Initial Release
V1.8	2016-12-18	Shi-Wan Lin (Thingswise), Mark Crawford (SAP) and Stephen Mellor (IIC).	1. This volume, containing the about roughly the same content as in Part I (from Chapter 1 to 7) of the initial release version, is released under the new IIC document structure. The content in Part II (from Chapter 8 to 16) will be published in separate volumes under the new IIC document structure. 2. A few new boilerplate sections (Chapter 1 and some appendices). 3. Improved and enhanced description of architecture concepts and constructs derived from ISO/IEC/IEEE 42010 Architecture Description standard and their application in IIRA (Chapter 3 Industrial Internet Architecture Framework). 4. Added a section on IIRA viewpoints' Scope of Applicability and Relationship to System Lifecycle Process (Chapter 3 Industrial Internet Architecture Framework, Section 3.6). 5. Added a section on Crosscutting Functions and System Characteristics (Chapter 6 Functional Viewpoint, Section 6.7). 6. Added a section on Functional Domain and Computational Deployment Models (Chapter 6 Functional Viewpoint, Section 6.8). 7. Added a section on Layered Databus Architecture Pattern (Chapter 7 Implementation Viewpoint, Section 7.5). 8. Added an appendix on Design Space Considerations (Appendix A).

V1.9	2019-05-xx	Shi-Wan Lin (Thingswise), Eric Simmon (NIST) and Stephen Mellor (IIC).	1. Added formal definitions from the Industrial Internet vocabulary for the Viewpoints and Functional Domains (across Chapter 4 Business View, Chapter 5 Usage View, Chapter 6 Functional View and Chapter 7 Implementation View). 2. Stated the kind of external influences, e.g. regulatory factors, that may need to be considered in conceptualizing an IIoT system (Chapter 4 Business Viewpoint). 3. Brought up the importance of wireless communication in the Control Domain (Chapter 6 Functional Domain, 6.3 The Control And Monitoring Domain). 4. Added the Human Roles in an IIoT System (Chapter 6 Functional View). 5. Clarified that the scope of the Implementation Viewpoint does not cover concerns about deployment and operation which are life-cycle issues to be considered separately (Chapter 7 Implementation View). 6. Clarified that the patterns are intended as examples and references (Chapter 7 Implementation View). 7. A number of editorial changes on sentences for better understanding.
V1.10	2022-09-15	Shi-Wan Lin (Thingswise), Eric Simmon (NIST) and Stephen Mellor (IIC).	1. Refined the reference architecture description so that it is better aligned with ISO/IEC/IEEE 42010, especially clarified the concepts of viewpoint, view and model kind. 2. Added a short guidance on using the IIRA. 3. Rewrite the content on Usage View to better reflect the usages in IIoT. 4. Revised the structures and contents in the functional domains, especially replacing Operation Domain with Management Domain, adjusting the content of information domain and restructured the Control Domain to better reflect the practices in IIoT systems. 5. Added the Functional Requirements Categories to map key IIoT functions to requirement categories for better understanding of the key functions. 6. Revised the text in the implementation view to emphasize the concept and usage of Architecture Patterns to better aid architects in designing their specific IIoT systems. 7. Added three important Architecture Patterns, IoT Component Capability Model Pattern, Digital Twin Core as a Middleware Architecture Pattern and System of Systems Orchestrator Architecture Pattern.

AUTHORS & LEGAL NOTICE

This document is a work product of the Industry IoT Consortium Architecture Task Group, co-chaired by Shi-Wan Lin (Thingswise) and Eric Simmon (NIST).

Authors: The following persons have written substantial portion of material content in this document:

Shi-Wan Lin (Thingswise/Intel, also co-editor), Eric Simmon (NIST, also co-editor), Daniel Young (Toshiba), Bradford Miller (GE), Jacques Durand (Fujitsu), Graham Bleakley (IBM), Amine Chigani (GE), Robert Martin (MITRE), Brett Murphy (RTI) and Mark Crawford (SAP).

Contributors: The following persons have contributed valuable ideas and feedback that significantly improve the content and quality of this document:

Marcellus Buchheit (Wibu-Systems), Anish Karmarkar (Oracle), Sven Schrecker (Intel), JP LeBlanc (Lynx Software Technologies), Chuck Byers (IIC), Hiroshi Yamamoto (Toshiba), Erin Bournival (Dell).

Technical Editor: Stephen Mellor (IIC staff) oversaw the process of organizing the contributions of the above Authors and Contributors into an integrated document.